Part A

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.

   Intrusion Detection System is a hardware or software monitor that detects attacks to a system or networks. Traditional intrusion detection system techniques make the system more complex and less efficient when dealing with Big Data, because its analysis properties process is complex and take a long time.

   Yes, the paper proposed to IDS classification method named Spark-Chi-SVM. First, in the data preprocessing stage, categorical data need to be converted to numerical data since SVM algorithm only process numerical data. Then standardize the dataset to avoid huge scale in the dataset. Second, ChiSqSelector method is used in the feature selection stage to determine the most important and most relevant features. Thirdly, use SVM model in Spark to classify the dataset. SVM could be hard to implement in machine learning, but using Spark significantly reduce the execution time.

2. Use python urllib library to extract the KDD Cup 99 data from their web repository, store it in a temporary location and then move it to the Databricks filesystem which can enable easy access to this data for analysis. Use the following commands in Databricks to get your data.

```
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	modificationTime
1	dbfs:/kdd/kddcup_data.gz	kddcup_data.gz	2144903	1667002942000

```
data_file = "dbfs:/kdd"

df = spark.read \
  .format("csv") \
  .option("inferSchema", True) \
  .option("header", False) \
  .option("sep", ',') \
  .option("path", data_file) \
  .load()
```

▸ (2) Spark Jobs

3. After storing the data into the Databricks filesystem. Load your data from the disk into Spark's RDD. Print 10 values of your RDD and verify the type of data structure of your data (RDD).

```
1  df.show(10)
```

(1) Spark Jobs

```
+---+---+----+---+---+----+---+---+---+----+----+----+----+----+----+----+----+----+----+----+----+----+----+----+----+----+----+----+----+----+----+----+----+----+----+----+-------+
|_c0|_c1| _c2|_c3|_c4| _c5|_c6|_c7|_c8|_c9|_c10|_c11|_c12|_c13|_c14|_c15|_c16|_c17|_c18|_c19|_c20|_c21|_c22|_c23|_c24|_c25|_c26|_c27|_c28|_c29|_c30|_c31|_c32|_c33|_c34|_c35|_c36|_c37|_c38|_c39|_c40|   _c41|
+---+---+----+---+---+----+---+---+---+----+----+----+----+----+----+----+----+----+----+----+----+----+----+----+----+----+----+----+----+----+----+----+----+----+----+----+-------+
|  0|tcp|http| SF|181|5450|  0|  0|  0|   0|   0|   1|   0|   0|   0|   0|   0|   0|   0|   0|   0|   0|   8|   8| 0.0| 0.0| 0.0| 0.0| 1.0| 0.0| 0.0|   9|   9| 1.0| 0.0|0.11| 0.0| 0.0| 0.0| 0.0| 0.0|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| 0.0| 0.0| 0.0| 0.0| 1.0| 0.0| 0.0|  19|  19| 1.0| 0.0|0.05| 0.0| 0.0| 0.0| 0.0| 0.0|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| 0.0| 0.0| 0.0| 0.0| 1.0| 0.0| 0.0|  29|  29| 1.0| 0.0|0.03| 0.0| 0.0| 0.0| 0.0| 0.0|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| 0.0| 0.0| 0.0| 0.0| 1.0| 0.0| 0.0|  39|  39| 1.0| 0.0|0.03| 0.0| 0.0| 0.0| 0.0| 0.0|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| 0.0| 0.0| 0.0| 0.0| 1.0| 0.0| 0.0|  49|  49| 1.0| 0.0|0.02| 0.0| 0.0| 0.0| 0.0| 0.0|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| 0.0| 0.0| 0.0| 0.0| 1.0| 0.0| 0.0|  59|  59| 1.0| 0.0|0.02| 0.0| 0.0| 0.0| 0.0| 0.0|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| 0.0| 0.0| 0.0| 0.0| 1.0| 0.0| 1.0|   1|  69| 1.0| 0.0| 1.0|0.04| 0.0| 0.0| 0.0| 0.0|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| 0.0| 0.0| 0.0| 0.0| 1.0| 0.0| 0.0|  11|  79| 1.0| 0.0|0.09|0.04| 0.0| 0.0| 0.0| 0.0|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| 0.0| 0.0| 0.0| 0.0| 1.0| 0.0| 0.0|   8|  89| 1.0| 0.0|0.12|0.04| 0.0| 0.0| 0.0| 0.0|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| 0.0| 0.0| 0.0| 0.0| 1.0| 0.0| 0.0|   8|  99| 1.0| 0.0|0.12|0.05| 0.0| 0.0| 0.0| 0.0|normal.|
+---+---+----+---+---+----+---+---+---+----+----+----+----+----+----+----+----+----+----+----+----+----+----+----+----+----+----+----+----+----+----+----+----+----+----+----+-------+
only showing top 10 rows
```

Command took 0.79 seconds -- by zqsteven.xie@mail.utoronto.ca at 10/29/2022, 8:03:12 PM on My Cluster

Cmd 4

```
1  df.schema
```

```
Out[6]: StructType(List(StructField(_c0,IntegerType,true),StructField(_c1,StringType,true),StructField(_c2,StringType,true),StructField(_c3,StringType,true),StructField(_c4,IntegerType,true),StructField(_c5,IntegerType,true),StructField(_c6,IntegerType,true),StructField(_c7,IntegerType,true),StructField(_c8,IntegerType,true),StructField(_c9,IntegerType,true),StructField(_c10,IntegerType,true),StructField(_c11,IntegerType,true),StructField(_c12,IntegerType,true),StructField(_c13,IntegerType,true),StructField(_c14,IntegerType,true),StructField(_c15,IntegerType,true),StructField(_c16,IntegerType,true),StructField(_c17,IntegerType,true),StructField(_c18,IntegerType,true),StructField(_c19,IntegerType,true),StructField(_c20,IntegerType,true),StructField(_c21,IntegerType,true),StructField(_c22,IntegerType,true),StructField(_c23,IntegerType,true),StructField(_c24,DoubleType,true),StructField(_c25,DoubleType,true),StructField(_c26,DoubleType,true),StructField(_c27,DoubleType,true),StructField(_c28,DoubleType,true),StructField(_c29,DoubleType,true),StructField(_c30,DoubleType,true),StructField(_c31,IntegerType,true),StructField(_c32,IntegerType,true),StructField(_c33,DoubleType,true),StructField(_c34,DoubleType,true),StructField(_c35,DoubleType,true),StructField(_c36,DoubleType,true),StructField(_c37,DoubleType,true),StructField(_c38,DoubleType,true),StructField(_c39,DoubleType,true),StructField(_c40,DoubleType,true),StructField(_c41,StringType,true)))
```

Command took 0.26 seconds -- by zqsteven.xie@mail.utoronto.ca at 10/29/2022, 8:06:46 PM on My Cluster

4. Split the data. (Each entry in your RDD is a comma-separated line of data, which you first need to split before you can parse and build your data frame.) Show the total number of features (columns) and print results.

```
1  df.show(10)
```

(1) Spark Jobs

```
+---+---+----+---+---+----+---+---+---+----+----+----+----+----+----+----+----+----+----+----+----+----+----+----+----+----+----+----+----+----+----+----+----+----+----+----+-------+
|_c0|_c1| _c2|_c3|_c4| _c5|_c6|_c7|_c8|_c9|_c10|_c11|_c12|_c13|_c14|_c15|_c16|_c17|_c18|_c19|_c20|_c21|_c22|_c23|_c24|_c25|_c26|_c27|_c28|_c29|_c30|_c31|_c32|_c33|_c34|_c35|_c36|_c37|_c38|_c39|_c40|   _c41|
+---+---+----+---+---+----+---+---+---+----+----+----+----+----+----+----+----+----+----+----+----+----+----+----+----+----+----+----+----+----+----+----+----+----+----+----+-------+
|  0|tcp|http| SF|181|5450|  0|  0|  0|   0|   0|   1|   0|   0|   0|   0|   0|   0|   0|   0|   0|   0|   8|   8| 0.0| 0.0| 0.0| 0.0| 1.0| 0.0| 0.0|   9|   9| 1.0| 0.0|0.11| 0.0| 0.0| 0.0| 0.0| 0.0|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| 0.0| 0.0| 0.0| 0.0| 1.0| 0.0| 0.0|  19|  19| 1.0| 0.0|0.05| 0.0| 0.0| 0.0| 0.0| 0.0|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| 0.0| 0.0| 0.0| 0.0| 1.0| 0.0| 0.0|  29|  29| 1.0| 0.0|0.03| 0.0| 0.0| 0.0| 0.0| 0.0|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| 0.0| 0.0| 0.0| 0.0| 1.0| 0.0| 0.0|  39|  39| 1.0| 0.0|0.03| 0.0| 0.0| 0.0| 0.0| 0.0|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| 0.0| 0.0| 0.0| 0.0| 1.0| 0.0| 0.0|  49|  49| 1.0| 0.0|0.02| 0.0| 0.0| 0.0| 0.0| 0.0|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| 0.0| 0.0| 0.0| 0.0| 1.0| 0.0| 0.0|  59|  59| 1.0| 0.0|0.02| 0.0| 0.0| 0.0| 0.0| 0.0|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| 0.0| 0.0| 0.0| 0.0| 1.0| 0.0| 1.0|   1|  69| 1.0| 0.0| 1.0|0.04| 0.0| 0.0| 0.0| 0.0|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| 0.0| 0.0| 0.0| 0.0| 1.0| 0.0| 0.0|  11|  79| 1.0| 0.0|0.09|0.04| 0.0| 0.0| 0.0| 0.0|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| 0.0| 0.0| 0.0| 0.0| 1.0| 0.0| 0.0|   8|  89| 1.0| 0.0|0.12|0.04| 0.0| 0.0| 0.0| 0.0|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| 0.0| 0.0| 0.0| 0.0| 1.0| 0.0| 0.0|   8|  99| 1.0| 0.0|0.12|0.05| 0.0| 0.0| 0.0| 0.0|normal.|
+---+---+----+---+---+----+---+---+---+----+----+----+----+----+----+----+----+----+----+----+----+----+----+----+----+----+----+----+----+----+----+----+----+----+----+----+-------+
only showing top 10 rows
```

5. Now extract these 6 columns (duration, protocol_type, service, src_bytes, dst_bytes, flag and label) from your dataset. Build a new RDD and dataframe. Print schema and display 10 values.

```
1  from pyspark.sql.functions import col
2  dff = df.select(col("_c0"),col("_c1"),col("_c2"),col("_c3"),col("_c4"),col("_c5"),)
```

Command took 0.09 seconds -- by zqsteven.xie@mail.utoronto.ca at 10/29/2022, 9:04:16 PM on My Cluster

Cmd 6

```
1  dff.show(10)
```

▸ (1) Spark Jobs

```
+---+---+----+---+---+----+
|_c0|_c1| _c2|_c3|_c4| _c5|
+---+---+----+---+---+----+
|  0|tcp|http| SF|181|5450|
|  0|tcp|http| SF|239| 486|
|  0|tcp|http| SF|235|1337|
|  0|tcp|http| SF|219|1337|
|  0|tcp|http| SF|217|2032|
|  0|tcp|http| SF|217|2032|
|  0|tcp|http| SF|212|1940|
|  0|tcp|http| SF|159|4087|
|  0|tcp|http| SF|210| 151|
|  0|tcp|http| SF|212| 786|
+---+---+----+---+---+----+
only showing top 10 rows
```

```
1  dff.schema
```

```
Out[20]: StructType(List(StructField(_c0,IntegerType,true),StructField(_c1,StringType,true),StructField(_
c2,StringType,true),StructField(_c3,StringType,true),StructField(_c4,IntegerType,true),StructField(_c5,In
tegerType,true)))
```

6. Get the total number of connections based on the protocol_type and based on the service. Show result in an ascending order. Plot the bar graph for both.

```
1  dff.groupBy((col("_c1")).alias("Protocal_Type")).count().sort("count", ascending = True).show()
```

▸ (2) Spark Jobs

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

```
1  p = dff.groupBy((col("_c1")).alias("Protocal_Type")).count().sort("count", ascending = True)
2  p_pandas = p.toPandas()
```

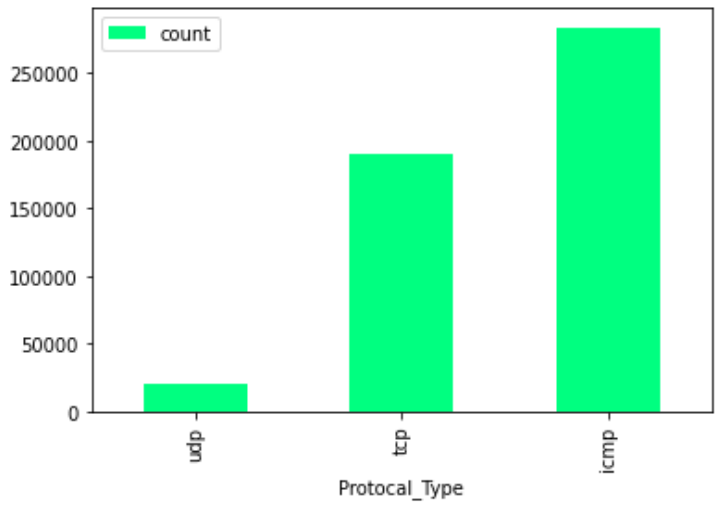
▸ (4) Spark Jobs

Command took 4.43 seconds -- by zqsteven.xie@mail.utoronto.ca at 10/29/2022, 9:56:00 PM on My Cluster

Cmd 11

```
1  p_pandas.plot(kind='bar', x = "Protocal_Type", y = 'count', colormap='winter_r')
```

Out[55]: <AxesSubplot:xlabel='Protocal_Type'>



```
pp = dff.groupBy((col("_c2")).alias("Service")).count().sort("count", ascending = True)
pp_pandas = pp.toPandas()
```

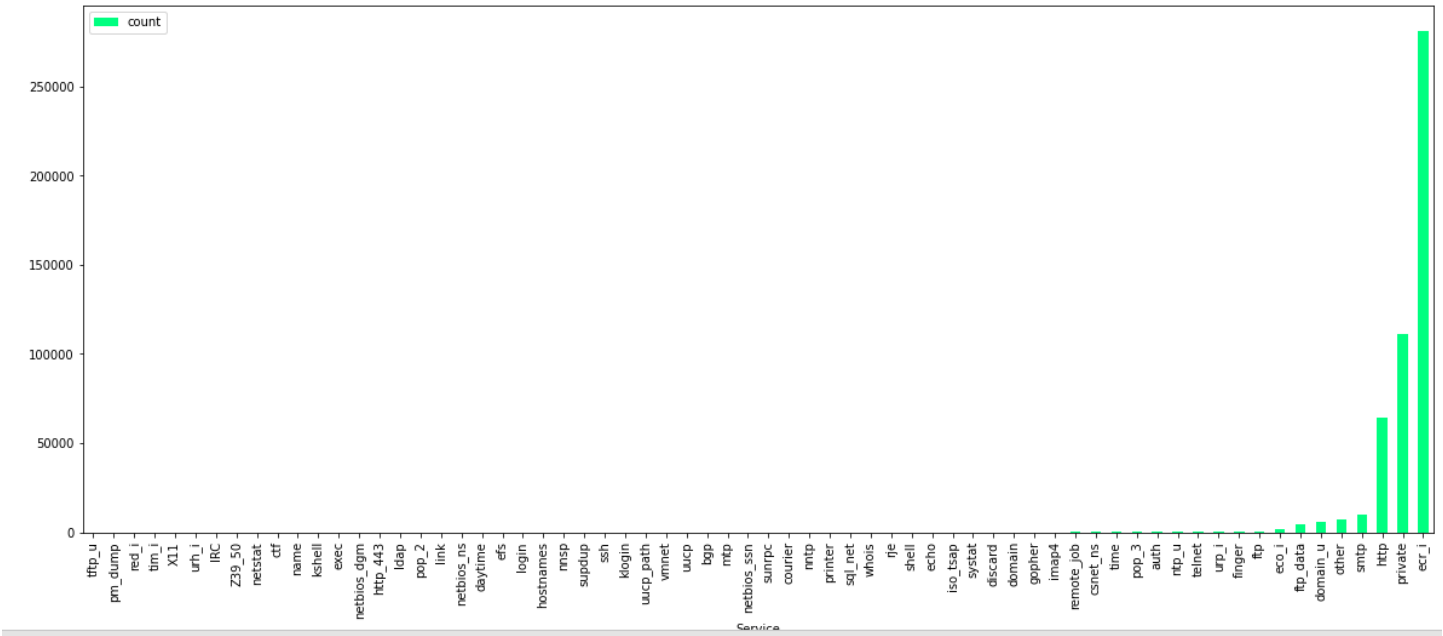
▸ (4) Spark Jobs

Command took 3.47 seconds -- by zqsteven.xie@mail.utoronto.ca at 10/30/2022, 8:04:36 PM on My Cluster

Cmd 19

```
import matplotlib.pyplot as plt
pp_pandas.plot(kind='bar', x = "Service", y = 'count', colormap='winter_r',figsize=(20, 8))
```

Out[18]: <AxesSubplot:xlabel='Service'>



7. Do a further exploratory data analysis, including other columns of this dataset and plot graphs. Plot at least 3 different charts and explain them.

   Explore the Service column _c2 and plot the pie chart of the top 5 service types.

```
p2 = dff.groupBy((col("_c2")).alias("Service Type")).count().sort("count", ascending = False)
p2.show(10)
```

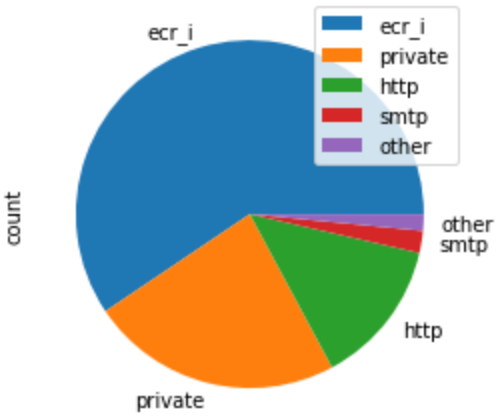
▸ (2) Spark Jobs

```
+------------+------+
|Service Type| count|
+------------+------+
|       ecr_i|281400|
|     private|110893|
|        http| 64293|
|        smtp|  9723|
|       other|  7237|
|    domain_u|  5863|
|    ftp_data|  4721|
|       eco_i|  1642|
|         ftp|   798|
|      finger|   670|
+------------+------+
only showing top 10 rows
```

```
p2_pandas = p2.toPandas()
p2_pandastop = p2_pandas[0:5]
p2_pandastop.plot(kind='pie', y = 'count', labels = p2_pandastop['Service Type'])
```

▸ (4) Spark Jobs

```
Out[63]: <AxesSubplot:ylabel='count'>
```



Explore the total number of data bytes from source to destination by different protocol types.

```
p3 = dff.groupBy((col("_c1")).alias("Protocal_Type")).sum('_c4')
p3.show()
```

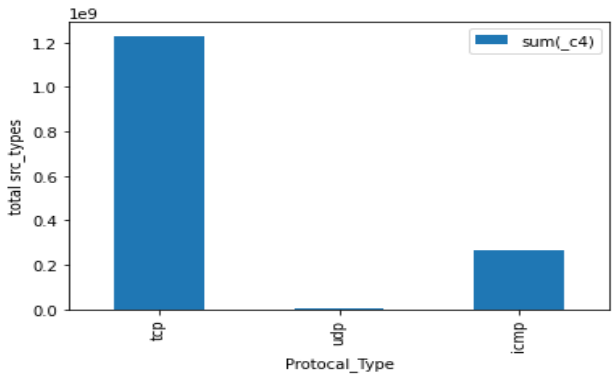
▸ (2) Spark Jobs

```
+-------------+----------+
|Protocal_Type|  sum(_c4)|
+-------------+----------+
|          tcp|1229530130|
|          udp|   1911953|
|         icmp| 263272941|
+-------------+----------+
```

```
import matplotlib.pyplot as plt
p3_pandas = p3.toPandas()
p3_pandas.plot(kind='bar', y='sum(_c4)',x='Protocal_Type')
plt.ylabel('total src_types')
```

▸ (2) Spark Jobs

```
Out[91]: Text(0, 0.5, 'total src_types')
```

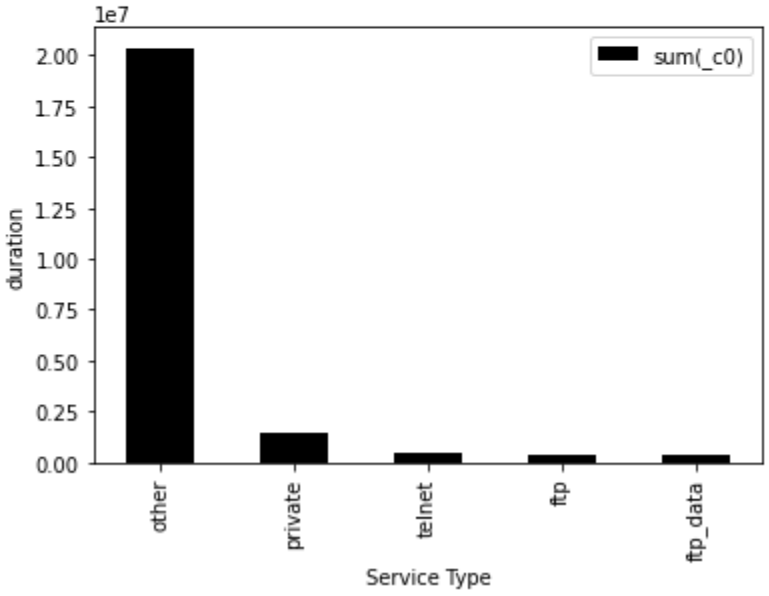


Explore the duration of connection by service types, plot the bar chart of top 5 service types of duration.

```
1  p4_pandas = p4.toPandas()
2  p4_pandastop = p4_pandas[0:5]
3  p4_pandastop.plot(kind='bar',y='sum(_c0)',x='Service Type',color = 'black')
4  plt.ylabel('duration')
```

▸ (4) Spark Jobs

```
Out[114]: Text(0, 0.5, 'duration')
```



8. Look at the label column where label == ‘normal’. Now create a new label column where you have a label == ‘normal’ and everything else is considered as an ‘attack’. Split your data (train/test) and based on your new label column now build a simple machine learning model for intrusion detection (you can use few selected columns for your model out of all). Explain which algorithm you have selected and why? Show the results with some success metrics.

Check the column where label is ‘normal.’, replace all that’s not ‘normal’ with ‘attack’.

```
dff.groupBy((col("_c41")).alias("Connection")).count().sort("count", ascending = False).show()
```

▸ (2) Spark Jobs

```
+----------------+------+
|      Connection| count|
+----------------+------+
|          smurf.|280790|
|        neptune.|107201|
|         normal.| 97278|
|           back.|  2203|
|          satan.|  1589|
|        ipsweep.|  1247|
|      portsweep.|  1040|
|    warezclient.|  1020|
|       teardrop.|   979|
|            pod.|   264|
|           nmap.|   231|
|   guess_passwd.|    53|
|buffer_overflow.|    30|
|           land.|    21|
|    warezmaster.|    20|
|           imap.|    12|
|        rootkit.|    10|
|     loadmodule.|     9|
```

```
from pyspark.sql.functions import when

targetdf = dff.withColumn("_c41", \
              when(df["_c41"] == 'normal.', 'normal').otherwise('attack'))
```

Command took 0.07 seconds -- by zqsteven.xie@mail.utoronto.ca at 10/30/2022, 9:04:38 AM on My Cluster

Cmd 21

```
targetdf.groupBy((col("_c41")).alias("Connection")).count().sort("count", ascending = False).show()
```

▸ (2) Spark Jobs

```
+----------+------+
|Connection| count|
+----------+------+
|    attack|396743|
|    normal| 97278|
+----------+------+
```

Replace normal labels by 0, and attack labels by 1 to use logistic regression algorithm for intrusion detection. Logistic regression analysis is a good estimation for predicting the likelihood of an event. It helps determine the probabilities between any two classes.

```
1  targetdf = dff.withColumn("_c41", \
2                when(df["_c41"] == 'normal.', 0).otherwise(1))
```

Command took 0.07 seconds -- by zqsteven.xie@mail.utoronto.ca at 10/30/2022,

Cmd 23

```
1  targetdf.show(10)
```

▸ (1) Spark Jobs

```
+---+---+----+---+----+---+----+
|_c0|_c1| _c2|_c4| _c5|_c6|_c41|
+---+---+----+---+----+---+----+
|  0|tcp|http|181|5450|  0|   0|
|  0|tcp|http|239| 486|  0|   0|
|  0|tcp|http|235|1337|  0|   0|
|  0|tcp|http|219|1337|  0|   0|
|  0|tcp|http|217|2032|  0|   0|
|  0|tcp|http|217|2032|  0|   0|
|  0|tcp|http|212|1940|  0|   0|
|  0|tcp|http|159|4087|  0|   0|
|  0|tcp|http|210| 151|  0|   0|
|  0|tcp|http|212| 786|  0|   0|
+---+---+----+---+----+---+----+
only showing top 10 rows
```

Preprocess the dataframe to divide the number columns and categorical columns and label columns to perform logistic regression in PySpark.

```
1  num_cols = ['_c0','_c4','_c5','_c6']
2  cat_cols = ['_c1','_c2']
3  label_col = '_c41'
```

Command took 0.03 seconds -- by zqsteven.xie@mail.utoronto.ca at 10/30/2022, 10:00:37 AM on My Cluster

Cmd 26

```
1  input_cols = num_cols
2
3  stages = []
4  for col in cat_cols:
5    string_indexer = StringIndexer(inputCol=col, outputCol=col + "Index")
6    stages += [string_indexer]
7    input_cols.append(col + "Index")
8  stages
```

Out[62]: [StringIndexer_c75e0ae3eeec, StringIndexer_715d6af6b46f]

Command took 0.07 seconds -- by zqsteven.xie@mail.utoronto.ca at 10/30/2022, 10:00:38 AM on My Cluster

Cmd 27

```
1  vect_assembler = VectorAssembler(inputCols= input_cols, outputCol="features")
2  stages += [vect_assembler]
3
4  pipeline = Pipeline().setStages(stages)
5  pipeline_model = pipeline.fit(targetdf)
6  train_df = pipeline_model.transform(targetdf)
```

```
display(train_df)
```

(1) Spark Jobs

Table +

	_c0	_c1	_c2	_c4	_c5	_c6	_c41	_c1Index	_c2Index	features
1	0	tcp	http	181	5450	0	0	1	2	{"vectorType": "dense", "length": 6, "values": [0, 181, 5450, 0, 1, 2]}
2	0	tcp	http	239	486	0	0	1	2	{"vectorType": "dense", "length": 6, "values": [0, 239, 486, 0, 1, 2]}
3	0	tcp	http	235	1337	0	0	1	2	{"vectorType": "dense", "length": 6, "values": [0, 235, 1337, 0, 1, 2]}
4	0	tcp	http	219	1337	0	0	1	2	{"vectorType": "dense", "length": 6, "values": [0, 219, 1337, 0, 1, 2]}
5	0	tcp	http	217	2032	0	0	1	2	{"vectorType": "dense", "length": 6, "values": [0, 217, 2032, 0, 1, 2]}
6	0	tcp	http	217	2032	0	0	1	2	{"vectorType": "dense", "length": 6, "values": [0, 217, 2032, 0, 1, 2]}
7	0	tcp	http	212	1940	0	0	1	2	{"vectorType": "dense", "length": 6, "values": [0, 212, 1940, 0, 1, 2]}

Split the data and perform logistic regression to the train data.

```
train, test = train_df.randomSplit([0.7, 0.3])
```

```
Command took 0.05 seconds -- by zqsteven.xie@mail.utoronto.ca at 10/30/2022, 10:00:48 AM on My Cluster
```

```
Cmd 30
```

```
from pyspark.ml.classification import LogisticRegression

# Create initial LogisticRegression model
lr = LogisticRegression(labelCol= label_col, featuresCol="features", maxIter=10)

# Train model with Training Data
lr_model = lr.fit(train)
```

Area under ROC and MSE evaluation metrics are used to see how well the algorithm performs.

```
from pyspark.ml.evaluation import BinaryClassificationEvaluator

test_pred = lr_model.transform(test)

evaluator_LR = BinaryClassificationEvaluator(rawPredictionCol="prediction",  labelCol=label_col,)
area_under_curve = evaluator_LR.evaluate(test_pred)

#default evaluation is areaUnderROC
print("areaUnderROC = %g" % area_under_curve)

evaluator_LR.getMetricName()
```

(3) Spark Jobs

```
areaUnderROC = 0.628641
Out[74]: 'areaUnderROC'
```

```
from pyspark.ml.evaluation import RegressionEvaluator
eva_MSE = RegressionEvaluator(metricName = "mse", predictionCol="prediction", labelCol=label_col)
pre = lr_model.transform(test)
mse = eva_MSE.evaluate(pre)
print(mse)
```

(1) Spark Jobs

```
0.1479511240126916
```

Part B

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.
   Answer: Yes, PaaS provides a framework that developers can build upon to develop or customize cloud-based applications. PaaS lets developers create applications using built-in software components. Platform as a Service components can give your development team new capabilities without your needing to add staff having the required skills.
   2. A platform as a service (PaaS) database offering in Azure provides built in high availability.
   Answer: Yes, Cloud features such as scalability, high-availability, and multi-tenant capability are included, reducing the amount of coding that developers must do.

2. A relational database must be used when:
   Answer: d. Strong consistency guarantees are required
   Relational database management systems use software that provides a consistent interface between users and applications and the database, making navigation much simpler for data users. This is particularly effective when working with big data since the volume of data dictates such consistency for users joining queries. Data consistency is a hallmark of relational database models since they maintain data integrity across applications and database copies.

3. When you are implementing a Software as a Service solution, you are responsible for:
   Answer: d. Configuring the SaaS solution
   Though SaaS application is usually developed with highly standardized software functionalities to serve as many clients as possible, many clients still ask for function variants according to their unique business needs through easy configuration and customization. SaaS vendors need take a well-designed strategy to enable self serve configuration and customization by their customers without changing the SaaS application source code for any individual customer.

4. 1. To achieve a hybrid cloud model, a company must always migrate from a private cloud model
   Answer: No. It is also possible to start with public cloud and then migrate part of the data to a on-premise infrastructure to achieve a hybrid cloud model.

   2. A company can extend the capacity of its internal network by using public cloud
   Answer: Yes, public cloud provides cloud environment data storage options which enables off-premise capacity.

   3. In a public cloud model, only guest users at your company can access the resources in the cloud
   No, permission of accessing the resources in the cloud can be given to anyone needed.

5. a. A cloud service that remains available after a failure occurs _____
   Answer: Fault tolerance, cloud service can run on multiple servers that keeps functioning when one of the servers fails.
   b. A cloud service that can be recovered after a failure occurs _____
   Answer: Disaster recovery, cloud service recovers from failures by replicating the data and storing it in a geographically different location.
   c. A cloud service that performs quickly when demand increases _____

Answer: Dynamic scalability, when the server on cloud is under heavy load, additional servers will add computational power to increase the capacity of computing.

d. A cloud service that can be accessed quickly from the internet _____

Answer: Low Latency, using cloud service saves the time and work load of local machines so it increases the response time.