Part A Question 2

In [0]:
```
#-------------------------------------------------#
# Part A Question 2:
print("Part A - Question 2:\n")
# Use python urllib library to extract the KDD Cup 99 data from their web repository
# and store it in a temporary location
import urllib.request
urllib.request.urlretrieve("http://kdd.ics.uci.edu/databases/kddcup99/kddcup.data_10
# move it to the Databricks filesystem
dbutils.fs.mv("file:/tmp/kddcup_data.gz", "dbfs:/kdd/kddcup_data.gz")
# Display
display(dbutils.fs.ls("dbfs:/kdd"))
```

```
Part A - Question 2:
```

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

Part A Question 3

In [0]:
```
#-------------------------------------------------#
# Part A - Question 3:
print("Part A - Question 3:\n")
#-------------------------------------------------#
from pyspark import SparkContext
sc = SparkContext.getOrCreate()
# Load data from the disk into Spark's RDD
data_file_path = "dbfs:/kdd/kddcup_data.gz"
data_set = sc.textFile(data_file_path)
#-------------------------------------------------#
# Print 10 values of RDD.
print("Print 10 values of RDD: \n")
#print(type(data_set.take(10)))
for one_line in data_set.take(10):
    print(one_line)
#-------------------------------------------------#
from pyspark.rdd import RDD
from pyspark.sql import DataFrame
# Verify the type of data structure of data.
print("\nIs the data structure RDD? Verification result: ", isinstance(data_set, RDD
data_set_RDD_orgn = data_set
del data_set
#-------------------------------------------------#
# Print important information of the dataset.
print("\nNumber of Enties in the data set: ", data_set_RDD_orgn.count())
print("Dimension of each entry: ", len(data_set_RDD_orgn.take(1)[0].split(",")))
```

```
Part A - Question 3:

Print 10 values of RDD:

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,
0.00,0.00,9,9,1.00,0.00,0.11,0.00,0.00,0.00,0.00,0.00,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,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,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,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.

Is the data structure RDD? Verification result:  True

Number of Enties in the data set:  494021
Dimension of each entry:  42
```

Part A Question 4

In [0]:

```
#-------------------------------------------------#
# Part A - Question 4:
print("Part A - Question 4: \n")
#-------------------------------------------------#
# Get the list of features from http://kdd.ics.uci.edu/databases/kddcup99/kddcup.nam
from urllib.request import urlopen
url = "http://kdd.ics.uci.edu/databases/kddcup99/kddcup.names"
feature_names_text = urlopen(url).read().decode("utf-8")
#-------------------------------------------------#
#print(feature_names_text)
feature_names_text = feature_names_text.replace(" ", "")
feature_names_list = [one_line.split(":")[0] for one_line in feature_names_text.spli
#-------------------------------------------------#
# Results commented here for potential use.
'''
feature_names_list = ['duration', 'protocol_type', 'service', 'flag', 'src_bytes', '
                      'num_failed_logins', 'logged_in', 'num_compromised', 'root_she
                      'num_shells', 'num_access_files', 'num_outbound_cmds', 'is_hos
                      'srv_serror_rate', 'rerror_rate', 'srv_rerror_rate', 'same_srv
                      'dst_host_srv_count', 'dst_host_same_srv_rate', 'dst_host_diff
                      'dst_host_serror_rate', 'dst_host_srv_serror_rate', 'dst_host_
                      '''
print("Number of features in the original data file: ", len(feature_names_list))
#-------------------------------------------------#
# Split the data.
print("Dimension of each entry (including label): ", len(data_set_RDD_orgn.take(1)[0
print("Number of Enties in the data set: ", data_set_RDD_orgn.count())
# data_set_list = [one_line.split(",") for one_line in data_set_RDD_orgn.take(data_s
#-------------------------------------------------#
# Split the data.
data_set_RDD = data_set_RDD_orgn.map(lambda line: line[:-1].split(","))
dataframe_titles = feature_names_list
dataframe_titles.append("label")
df_KDDCup = data_set_RDD.toDF(dataframe_titles)
print("\nThe schema is of the original dataset is")
df_KDDCup.printSchema()
#df_KDDCup.show(truncate=False)
print("Print one record of the dataframe: ")
```

```
df_KDDCup.show(n=1, truncate=False, vertical=True)
print("\nThe total number of columns (including label) is: ", len(df_KDDCup.columns))
```

```
Part A - Question 4:

Number of features in the original data file:  41
Dimension of each entry (including label):  42
Number of Enties in the data set:  494021

The schema is of the original dataset is
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)
 |-- is_host_login: string (nullable = true)
 |-- is_guest_login: string (nullable = true)
 |-- count: string (nullable = true)
 |-- srv_count: string (nullable = true)
 |-- serror_rate: string (nullable = true)
 |-- srv_serror_rate: string (nullable = true)
 |-- rerror_rate: string (nullable = true)
 |-- srv_rerror_rate: string (nullable = true)
 |-- same_srv_rate: string (nullable = true)
 |-- diff_srv_rate: string (nullable = true)
 |-- srv_diff_host_rate: string (nullable = true)
 |-- dst_host_count: string (nullable = true)
 |-- dst_host_srv_count: string (nullable = true)
 |-- dst_host_same_srv_rate: string (nullable = true)
 |-- dst_host_diff_srv_rate: string (nullable = true)
 |-- dst_host_same_src_port_rate: string (nullable = true)
 |-- dst_host_srv_diff_host_rate: string (nullable = true)
 |-- dst_host_serror_rate: string (nullable = true)
 |-- dst_host_srv_serror_rate: string (nullable = true)
 |-- dst_host_rerror_rate: string (nullable = true)
 |-- dst_host_srv_rerror_rate: string (nullable = true)
 |-- label: string (nullable = true)

Print one record of the dataframe:
-RECORD 0-----------------------------
 duration                     | 0
 protocol_type                | tcp
 service                      | http
 flag                         | SF
 src_bytes                    | 181
 dst_bytes                    | 5450
 land                         | 0
 wrong_fragment               | 0
 urgent                       | 0
 hot                          | 0
 num_failed_logins            | 0
 logged_in                    | 1
 num_compromised              | 0
```

```
 root_shell                   | 0
 su_attempted                 | 0
 num_root                     | 0
 num_file_creations           | 0
 num_shells                   | 0
 num_access_files             | 0
 num_outbound_cmds            | 0
 is_host_login                | 0
 is_guest_login               | 0
 count                        | 8
 srv_count                    | 8
 serror_rate                  | 0.00
 srv_serror_rate              | 0.00
 rerror_rate                  | 0.00
 srv_rerror_rate              | 0.00
 same_srv_rate                | 1.00
 diff_srv_rate                | 0.00
 srv_diff_host_rate           | 0.00
 dst_host_count               | 9
 dst_host_srv_count           | 9
 dst_host_same_srv_rate       | 1.00
 dst_host_diff_srv_rate       | 0.00
 dst_host_same_src_port_rate  | 0.11
 dst_host_srv_diff_host_rate  | 0.00
 dst_host_serror_rate         | 0.00
 dst_host_srv_serror_rate     | 0.00
 dst_host_rerror_rate         | 0.00
 dst_host_srv_rerror_rate     | 0.00
 label                        | normal
only showing top 1 row


The total number of columns (including label) is:  42
```

Part A Question 5

In [0]:

```
#-----------------------------------------------------#
# Part A - Question 5:
print("Part A - Question 5: \n")
#-----------------------------------------------------#
# Extract these 6 columns (duration, protocol_type, service, src_bytes, dst_bytes, f
df_KDDCup_extract = df_KDDCup["duration", "protocol_type", "service", "src_bytes", "
print("\nThe schema is of the extracted dataframe is")
df_KDDCup_extract.printSchema()
#-----------------------------------------------------#
# Build a new dataframe w/ extracted features
print("Print one record of the extracted dataframe (shown vertically): ")
df_KDDCup_extract.show(n=1, truncate=False, vertical=True)
print("Print first several records of the extracted dataframe: ")
df_KDDCup_extract.show(n=10, truncate=False)
print("\nThe total number of columns (including label) is: ", len(df_KDDCup_extract.
#-----------------------------------------------------#
# Build a new RDD
RDD_KDDCup_extracted = df_KDDCup_extract.rdd
print("\nPrint first ten rows of the newly created RDD: ")
for one_row in RDD_KDDCup_extracted.take(10):
    print(one_row)
```

```
Part A - Question 5:


The schema is of the extracted dataframe is
root
 |-- duration: string (nullable = true)
 |-- protocol_type: string (nullable = true)
```

```
 |-- service: string (nullable = true)
 |-- src_bytes: string (nullable = true)
 |-- dst_bytes: string (nullable = true)
 |-- flag: string (nullable = true)
 |-- label: string (nullable = true)

Print one record of the extracted dataframe (shown vertically):
-RECORD 0---------------
 duration      | 0
 protocol_type | tcp
 service       | http
 src_bytes     | 181
 dst_bytes     | 5450
 flag          | SF
 label         | normal
only showing top 1 row

Print first several records of the extracted dataframe:
+--------+-------------+-------+---------+---------+----+------+
|duration|protocol_type|service|src_bytes|dst_bytes|flag|label |
+--------+-------------+-------+---------+---------+----+------+
|0       |tcp          |http   |181      |5450     |SF  |normal|
|0       |tcp          |http   |239      |486      |SF  |normal|
|0       |tcp          |http   |235      |1337     |SF  |normal|
|0       |tcp          |http   |219      |1337     |SF  |normal|
|0       |tcp          |http   |217      |2032     |SF  |normal|
|0       |tcp          |http   |217      |2032     |SF  |normal|
|0       |tcp          |http   |212      |1940     |SF  |normal|
|0       |tcp          |http   |159      |4087     |SF  |normal|
|0       |tcp          |http   |210      |151      |SF  |normal|
|0       |tcp          |http   |212      |786      |SF  |normal|
+--------+-------------+-------+---------+---------+----+------+
only showing top 10 rows


The total number of columns (including label) is:  7

Print first ten rows of the newly created RDD:
Row(duration='0', protocol_type='tcp', service='http', src_bytes='181', dst_bytes='5
450', flag='SF', label='normal')
Row(duration='0', protocol_type='tcp', service='http', src_bytes='239', dst_bytes='4
86', flag='SF', label='normal')
Row(duration='0', protocol_type='tcp', service='http', src_bytes='235', dst_bytes='1
337', flag='SF', label='normal')
Row(duration='0', protocol_type='tcp', service='http', src_bytes='219', dst_bytes='1
337', flag='SF', label='normal')
Row(duration='0', protocol_type='tcp', service='http', src_bytes='217', dst_bytes='2
032', flag='SF', label='normal')
Row(duration='0', protocol_type='tcp', service='http', src_bytes='217', dst_bytes='2
032', flag='SF', label='normal')
Row(duration='0', protocol_type='tcp', service='http', src_bytes='212', dst_bytes='1
940', flag='SF', label='normal')
Row(duration='0', protocol_type='tcp', service='http', src_bytes='159', dst_bytes='4
087', flag='SF', label='normal')
Row(duration='0', protocol_type='tcp', service='http', src_bytes='210', dst_bytes='1
51', flag='SF', label='normal')
Row(duration='0', protocol_type='tcp', service='http', src_bytes='212', dst_bytes='7
86', flag='SF', label='normal')
```

Part A - Question 6

In [0]:

```
#-------------------------------------------------#
# Part A - Question 6:
print("Part A - Question 6: \n")
#-------------------------------------------------#
print("\nGet the total number of connections based on the protocol_type: ")
```

```
DF_connection_protocol = df_KDDCup_extract.groupBy("protocol_type").count().sort("co
display(DF_connection_protocol)
DF_connection_protocol.show(truncate=False)
```

```
Part A - Question 6:


Get the total number of connections based on the protocol_type:
+-------------+------+
|protocol_type|count |
+-------------+------+
|udp          |20354 |
|tcp          |190065|
|icmp         |283602|
+-------------+------+
```

protocol_type	count
udp	20354
tcp	190065
icmp	283602

In [0]:

```
#-------------------------------------------------#
print("\nGet the total number of connections based on the service: ")
DF_connection_service = df_KDDCup_extract.groupBy("service").count().sort("count")
display(DF_connection_service)
DF_connection_service.show(truncate=False)
```

```
Get the total number of connections based on the service:
+-----------+-----+
|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   |
|netbios_dgm|99   |
|http_443   |99   |
|exec       |99   |
|ldap       |101  |
|pop_2      |101  |
|link       |102  |
|netbios_ns |102  |
|daytime    |103  |
+-----------+-----+
only showing top 20 rows
```

service	count
tftp_u	1
red_i	1
pm_dump	1

service	count
tim_i	7
X11	11
urh_i	14
IRC	43
Z39_50	92

Part A - Question 7:

In [0]:
```
#-------------------------------------------------#
# Part A - Question 7:
print("Part A - Question 7: \n")
#-------------------------------------------------#
print("\nGet the total number of connections based on the label: ")
DF_connection_analysis = df_KDDCup_extract.groupBy("label").count().orderBy("count",
display(DF_connection_analysis)
DF_connection_analysis.show(truncate=False)
#-------------------------------------------------#
```

```
Part A - Question 7:


Get the total number of connections based on the label:
+---------------+------+
|label          |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     |
|ftp_write      |8     |
|multihop       |7     |
+---------------+------+
only showing top 20 rows
```

label	count
smurf	280790
neptune	107201
normal	97278
back	2203
satan	1589

label	count
ipsweep	1247
portsweep	1040
warezclient	1020

In [0]:
```
#-------------------------------------------------#
print("\nGet the total number of normal connections based on the protocol type and c
DF_connection_analysis = df_KDDCup_extract.filter(df_KDDCup_extract.label == "normal
display(DF_connection_protocol)
display(DF_connection_analysis)
print("\nTtotal number of connections based on the protocol type: ")
DF_connection_protocol.show(truncate=False)
print("\nTtotal number of normal connections based on the protocol type: ")
DF_connection_analysis.show(truncate=False)
```

```
Get the total number of normal connections based on the protocol type and compare wi
th all types of connections.

Ttotal number of connections based on the protocol type:
+-------------+------+
|protocol_type|count |
+-------------+------+
|udp          |20354 |
|tcp          |190065|
|icmp         |283602|
+-------------+------+


Ttotal number of normal connections based on the protocol type:
+-------------+-----+
|protocol_type|count|
+-------------+-----+
|icmp         |1288 |
|udp          |19177|
|tcp          |76813|
+-------------+-----+
```

protocol_type	count
udp	20354
tcp	190065
icmp	283602

protocol_type	count
icmp	1288
udp	19177
tcp	76813

In [0]:
```
#-------------------------------------------------#
print("\nGet the total number of normal connections based on the service and compare
DF_connection_analysis = df_KDDCup_extract.filter(df_KDDCup_extract.label == "normal
display(DF_connection_service.orderBy("count", ascending=False))
display(DF_connection_analysis)
print("\nTtotal number of connections based on the service: ")
```

```
DF_connection_service.orderBy("count", ascending=False).show(truncate=False)
print("\nTtotal number of normal connections based on the service: ")
DF_connection_analysis.show(truncate=False)
```

```
Get the total number of normal connections based on the service and compare with all
types of connections.

Ttotal number of connections based on the service:
+----------+------+
|service   |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   |
|urp_i     |538   |
|telnet    |513   |
|ntp_u     |380   |
|auth      |328   |
|pop_3     |202   |
|time      |157   |
|csnet_ns  |126   |
|remote_job|120   |
|gopher    |117   |
|imap4     |117   |
+----------+------+
only showing top 20 rows


Ttotal number of normal connections based on the service:
+--------+-----+
|service |count|
+--------+-----+
|http    |61886|
|smtp    |9598 |
|private |7366 |
|domain_u|5862 |
|other   |5632 |
|ftp_data|3798 |
|urp_i   |537  |
|finger  |468  |
|eco_i   |389  |
|ntp_u   |380  |
|ftp     |373  |
|ecr_i   |345  |
|auth    |220  |
|telnet  |219  |
|pop_3   |79   |
|time    |52   |
|IRC     |42   |
|urh_i   |14   |
|X11     |9    |
|domain  |3    |
+--------+-----+
only showing top 20 rows
```

service	count
ecr_i	281400
private	110893

service	count
http	64293
smtp	9723
other	7237
domain_u	5863
ftp_data	4721
eco_i	1642

service	count
http	61886
smtp	9598
private	7366
domain_u	5862
other	5632
ftp_data	3798
urp_i	537
finger	468
eco_i	389
ntp_u	380

Part A - Question 8:

In [2]:

```
#-------------------------------------------------#
# Part A - Question 8:
print("Part A - Question 8: \n")

#-------------------------------------------------#
from pyspark.mllib.classification import SVMWithSGD, SVMModel
from pyspark.ml.evaluation import MulticlassClassificationEvaluator
from pyspark.sql.functions import when
from pyspark.sql.functions import lit

df_KDDCup_extract = df_KDDCup_extract.withColumn("label_2", when((df_KDDCup_extract.
display(df_KDDCup_extract)


# Load and parse the data
def parsePoint(line):
    return LabeledPoint(values[0], values[1:])
RDD_KDDCup_ML = RDD_KDDCup_extracted.map(parsePoint)


splits = RDD_KDDCup_ML.randomSplit([0.7,0.3])
train_set = splits[0]
test_set = splits[1]
clf = SVMWithSGD()

model_fit = clf.train(train_set)

# Further Analysis is not shown here due to unstability of databricks.
```

Part A - Question 8: