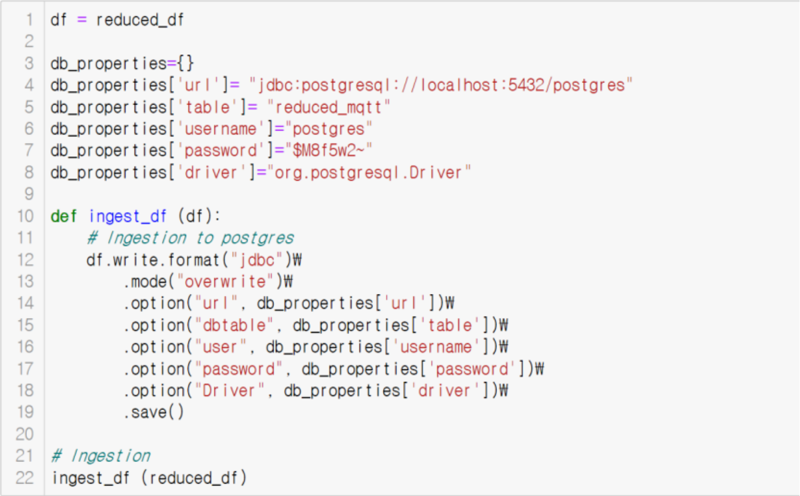
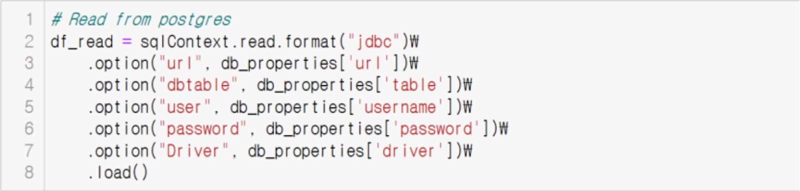
Task I

- Ingestion & Reading Code

Postgres

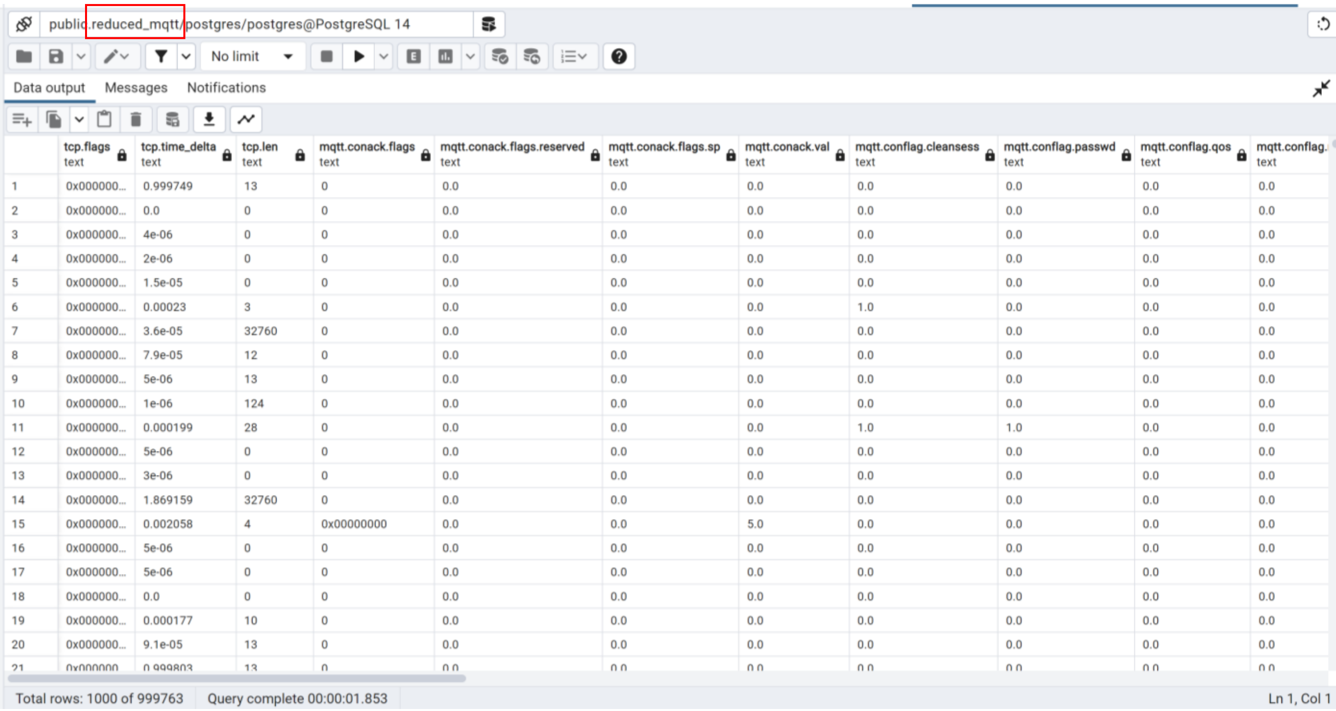


```
df = reduced_df

db_properties={}
db_properties['url']= "jdbc:postgresql://localhost:5432/postgres"
db_properties['table']= "reduced_mqtt"
db_properties['username']="postgres"
db_properties['password']="$M8f5w2~"
db_properties['driver']="org.postgresql.Driver"

def ingest_df (df):
    # Ingestion to postgres
    df.write.format("jdbc")₩
        .mode("overwrite")₩
        .option("url", db_properties['url'])₩
        .option("dbtable", db_properties['table'])₩
        .option("user", db_properties['username'])₩
        .option("password", db_properties['password'])₩
        .option("Driver", db_properties['driver'])₩
        .save()

# Ingestion
ingest_df (reduced_df)
```



```
# Read from postgres
df_read = sqlContext.read.format("jdbc")₩
    .option("url", db_properties['url'])₩
    .option("dbtable", db_properties['table'])₩
    .option("user", db_properties['username'])₩
    .option("password", db_properties['password'])₩
    .option("Driver", db_properties['driver'])₩
    .load()
```

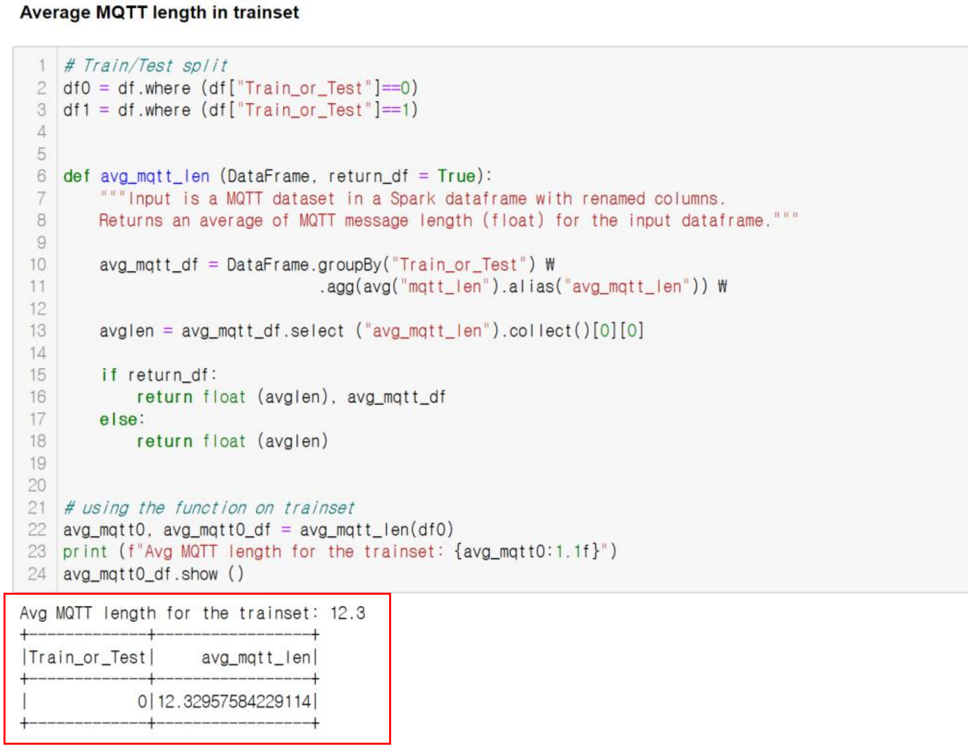
- Table in PG Admin4 (Reduced dataset)



	tcp.flags	tcp.time_delta	tcp.len	mqtt.conack.flags	mqtt.conack.flags.reserved	mqtt.conack.flags.sp	mqtt.conack.val	mqtt.conflag.cleansess	mqtt.conflag.passwd	mqtt.conflag.qos	mqtt.conflag.
	text	text	text	text	text	text	text	text	text	text	text
1	0x000000...	0.999749	13	0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
2	0x000000...	0.0	0	0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
3	0x000000...	4e-06	0	0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
4	0x000000...	2e-06	0	0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
5	0x000000...	1.5e-05	0	0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
6	0x000000...	0.00023	3	0	0.0	0.0	0.0	1.0	0.0	0.0	0.0
7	0x000000...	3.6e-05	32760	0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
8	0x000000...	7.9e-05	12	0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
9	0x000000...	5e-06	13	0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
10	0x000000...	1e-06	124	0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
11	0x000000...	0.000199	28	0	0.0	0.0	0.0	1.0	1.0	0.0	0.0
12	0x000000...	5e-06	0	0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
13	0x000000...	3e-06	0	0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
14	0x000000...	1.869159	32760	0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
15	0x000000...	0.002058	4	0x00000000	0.0	0.0	5.0	0.0	0.0	0.0	0.0
16	0x000000...	5e-06	0	0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
17	0x000000...	5e-06	0	0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
18	0x000000...	0.0	0	0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
19	0x000000...	0.000177	10	0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
20	0x000000...	9.1e-05	13	0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
21	0x000000	0.999803	13	0	0.0	0.0	0.0	0.0	0.0	0.0	0.0

Total rows: 1000 of 999763 Query complete 00:00:01.853 Ln 1, Col 1

TASK II-1

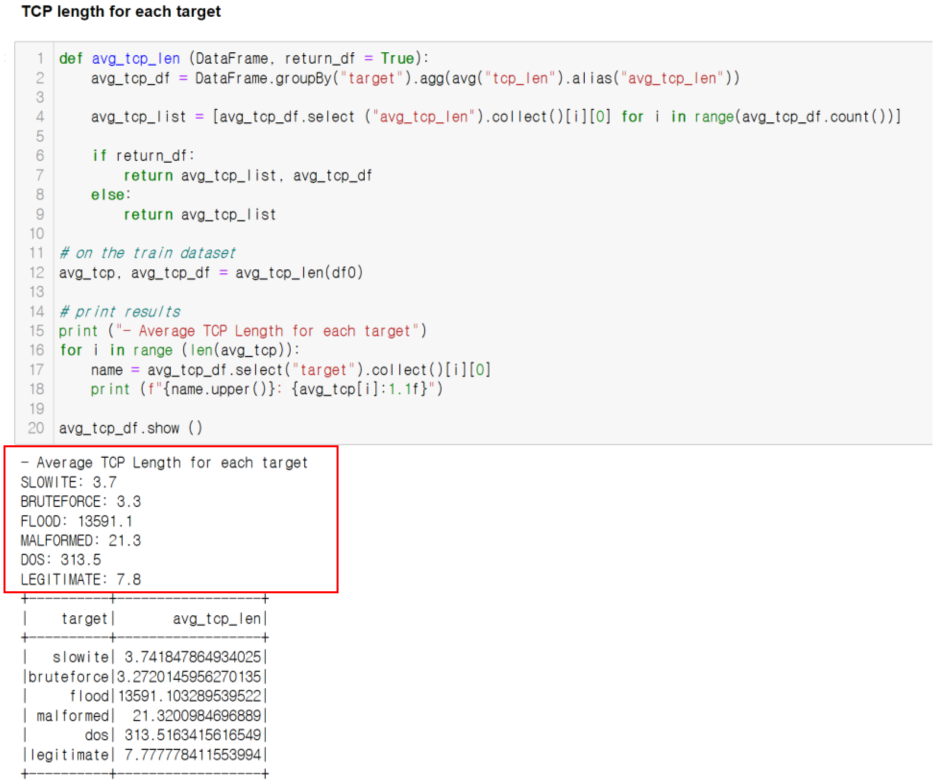


Average MQTT length in trainset

```
# Train/Test split
df0 = df.where (df["Train_or_Test"]==0)
df1 = df.where (df["Train_or_Test"]==1)


def avg_mqtt_len (DataFrame, return_df = True):
    """Input is a MQTT dataset in a Spark dataframe with renamed columns.
    Returns an average of MQTT message length (float) for the input dataframe."""

    avg_mqtt_df = DataFrame.groupBy("Train_or_Test") ₩
                        .agg(avg("mqtt_len").alias("avg_mqtt_len")) ₩

    avglen = avg_mqtt_df.select ("avg_mqtt_len").collect()[0][0]

    if return_df:
        return float (avglen), avg_mqtt_df
    else:
        return float (avglen)


# using the function on trainset
avg_mqtt0, avg_mqtt0_df = avg_mqtt_len(df0)
print (f"Avg MQTT length for the trainset: {avg_mqtt0:1.1f}")
avg_mqtt0_df.show ()
```

```
Avg MQTT length for the trainset: 12.3
+-------------+-----------------+
|Train_or_Test|     avg_mqtt_len|
+-------------+-----------------+
|            0|12.32957584229114|
+-------------+-----------------+
```

TASK II-2

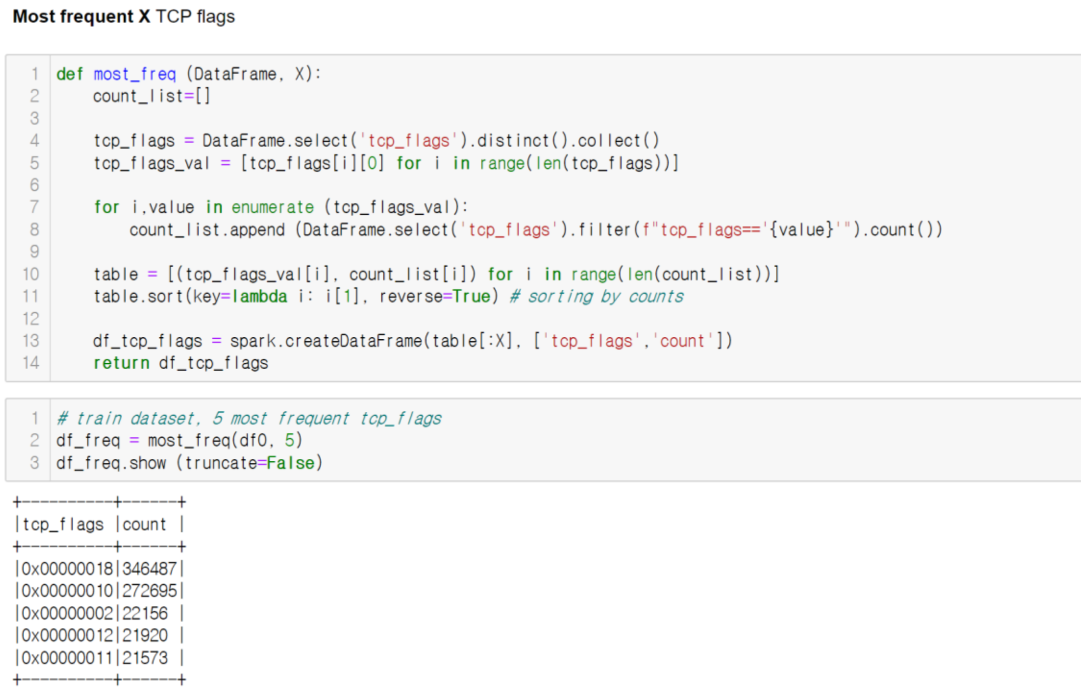


TCP length for each target

```
def avg_tcp_len (DataFrame, return_df = True):
    avg_tcp_df = DataFrame.groupBy("target").agg(avg("tcp_len").alias("avg_tcp_len"))

    avg_tcp_list = [avg_tcp_df.select ("avg_tcp_len").collect()[i][0] for i in range(avg_tcp_df.count())]

    if return_df:
        return avg_tcp_list, avg_tcp_df
    else:
        return avg_tcp_list

# on the train dataset
avg_tcp, avg_tcp_df = avg_tcp_len(df0)

# print results
print ("- Average TCP Length for each target")
for i in range (len(avg_tcp)):
    name = avg_tcp_df.select("target").collect()[i][0]
    print (f"{name.upper()}: {avg_tcp[i]:1.1f}")

avg_tcp_df.show ()
```

```
- Average TCP Length for each target
SLOWITE: 3.7
BRUTEFORCE: 3.3
FLOOD: 13591.1
MALFORMED: 21.3
DOS: 313.5
LEGITIMATE: 7.8
+----------+------------------+
|    target|       avg_tcp_len|
+----------+------------------+
|   slowite| 3.741847864934025|
|bruteforce|3.2720145956270135|
|     flood|13591.103289539522|
| malformed|  21.3200984696889|
|       dos| 313.5163415616549|
|legitimate| 7.777778411553994|
+----------+------------------+
```

TASK II-3



Most frequent X TCP flags

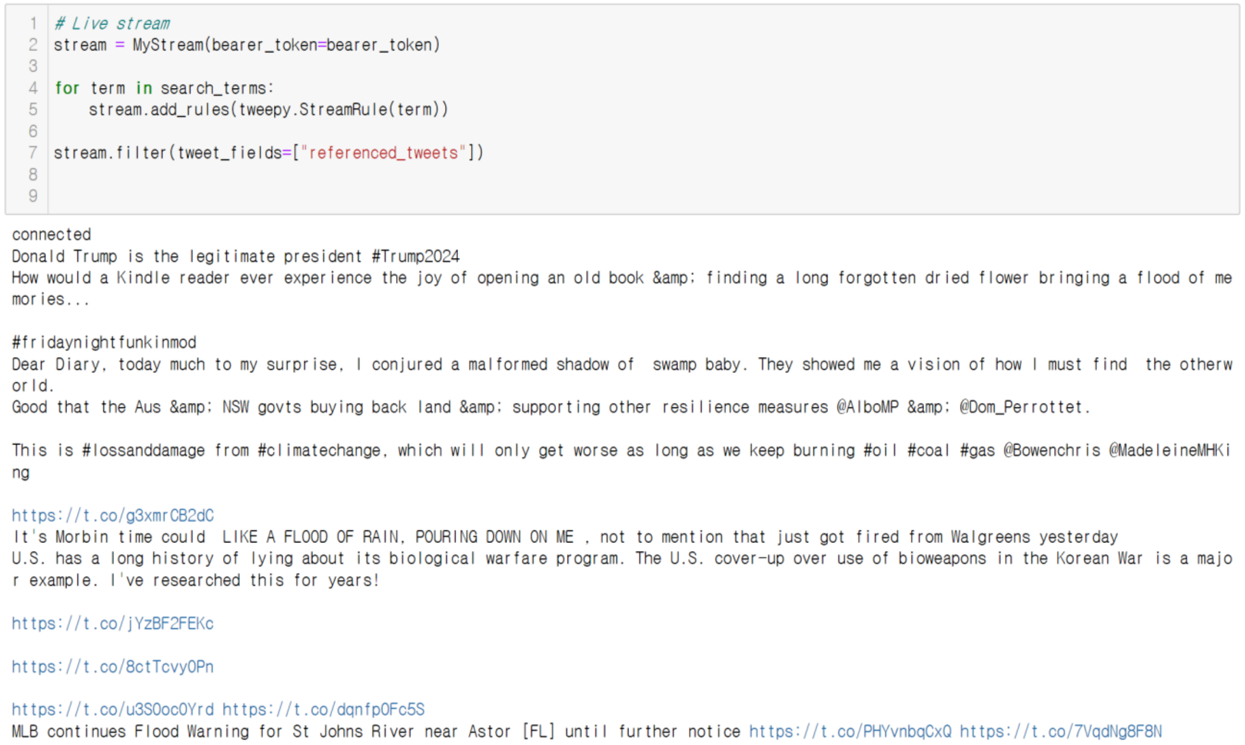
```python
def most_freq (DataFrame, X):
    count_list=[]

    tcp_flags = DataFrame.select('tcp_flags').distinct().collect()
    tcp_flags_val = [tcp_flags[i][0] for i in range(len(tcp_flags))]

    for i,value in enumerate (tcp_flags_val):
        count_list.append (DataFrame.select('tcp_flags').filter(f"tcp_flags=='{value}'").count())

    table = [(tcp_flags_val[i], count_list[i]) for i in range(len(count_list))]
    table.sort(key=lambda i: i[1], reverse=True) # sorting by counts

    df_tcp_flags = spark.createDataFrame(table[:X], ['tcp_flags','count'])
    return df_tcp_flags
```

```python
# train dataset, 5 most frequent tcp_flags
df_freq = most_freq(df0, 5)
df_freq.show (truncate=False)
```

```
+----------+------+
|tcp_flags |count |
+----------+------+
|0x00000018|346487|
|0x00000010|272695|
|0x00000002|22156 |
|0x00000012|21920 |
|0x00000011|21573 |
+----------+------+
```

TASK II-4: Twitter feed

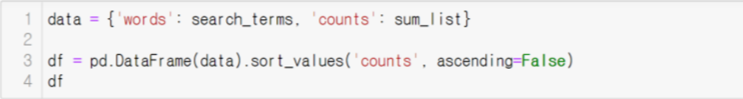
- Apache Kafka Streaming



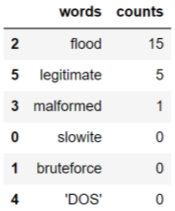
```python
# Live stream
stream = MyStream(bearer_token=bearer_token)

for term in search_terms:
    stream.add_rules(tweepy.StreamRule(term))

stream.filter(tweet_fields=["referenced_tweets"])


```

```
connected
Donald Trump is the legitimate president #Trump2024
How would a Kindle reader ever experience the joy of opening an old book &amp; finding a long forgotten dried flower bringing a flood of me
mories...

#fridaynightfunkinmod
Dear Diary, today much to my surprise, I conjured a malformed shadow of  swamp baby. They showed me a vision of how I must find  the otherw
orld.
Good that the Aus &amp; NSW govts buying back land &amp; supporting other resilience measures @AlboMP &amp; @Dom_Perrottet.

This is #lossanddamage from #climatechange, which will only get worse as long as we keep burning #oil #coal #gas @Bowenchris @MadeleineMHKi
ng

https://t.co/g3xmrCB2dC
It's Morbin time could  LIKE A FLOOD OF RAIN, POURING DOWN ON ME , not to mention that just got fired from Walgreens yesterday
U.S. has a long history of lying about its biological warfare program. The U.S. cover-up over use of bioweapons in the Korean War is a majo
r example. I've researched this for years!

https://t.co/jYzBF2FEKc

https://t.co/8ctTcvy0Pn

https://t.co/u3S0oc0Yrd https://t.co/dqnfp0Fc5S
MLB continues Flood Warning for St Johns River near Astor [FL] until further notice https://t.co/PHYvnbqCxQ https://t.co/7VqdNg8F8N
```

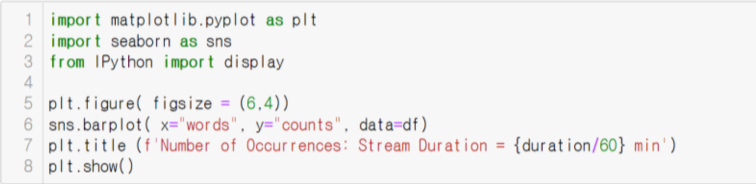
- Result with 5 min feed



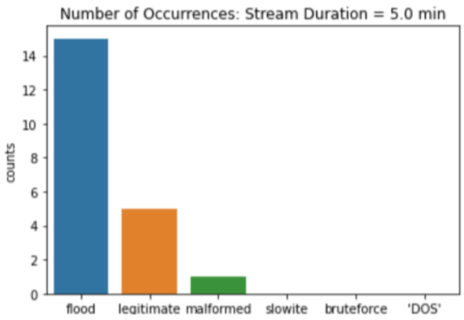
```
data = {'words': search_terms, 'counts': sum_list}

df = pd.DataFrame(data).sort_values('counts', ascending=False)
df
```



	words	counts
2	flood	15
5	legitimate	5
3	malformed	1
0	slowite	0
1	bruteforce	0
4	'DOS'	0



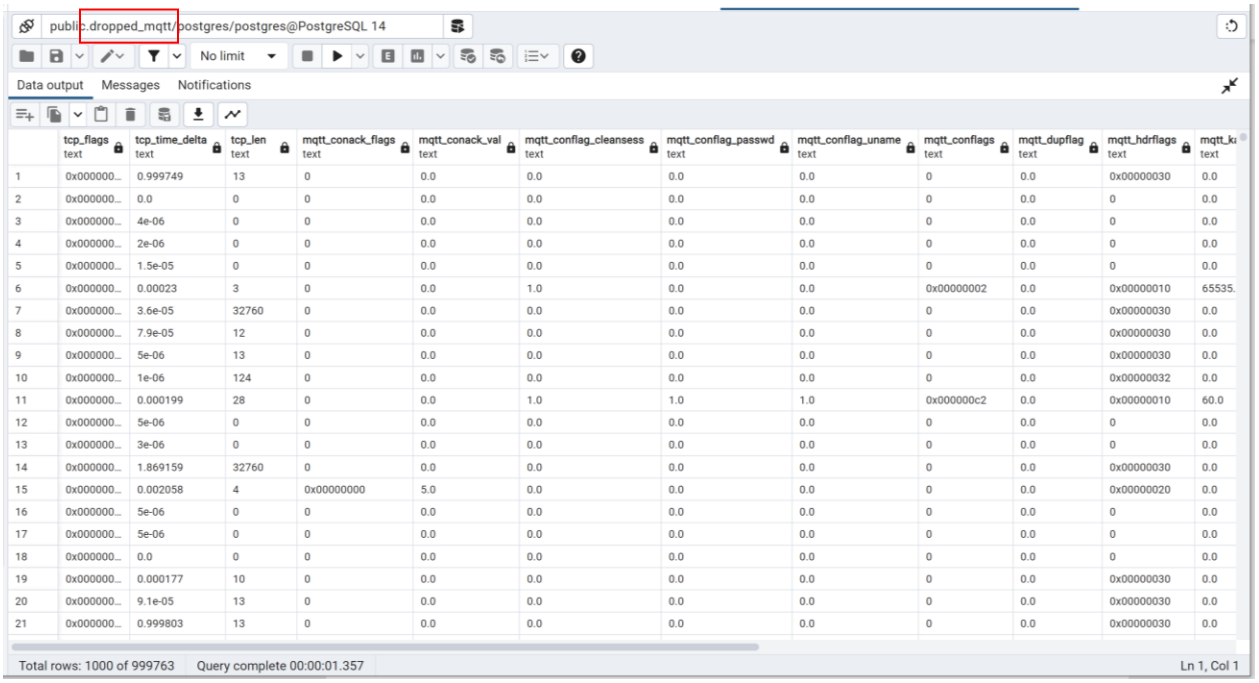
```
import matplotlib.pyplot as plt
import seaborn as sns
from IPython import display

plt.figure( figsize = (6,4))
sns.barplot( x="words", y="counts", data=df)
plt.title (f'Number of Occurrences: Stream Duration = {duration/60} min')
plt.show()
```



TASK III

Data Preprocessing

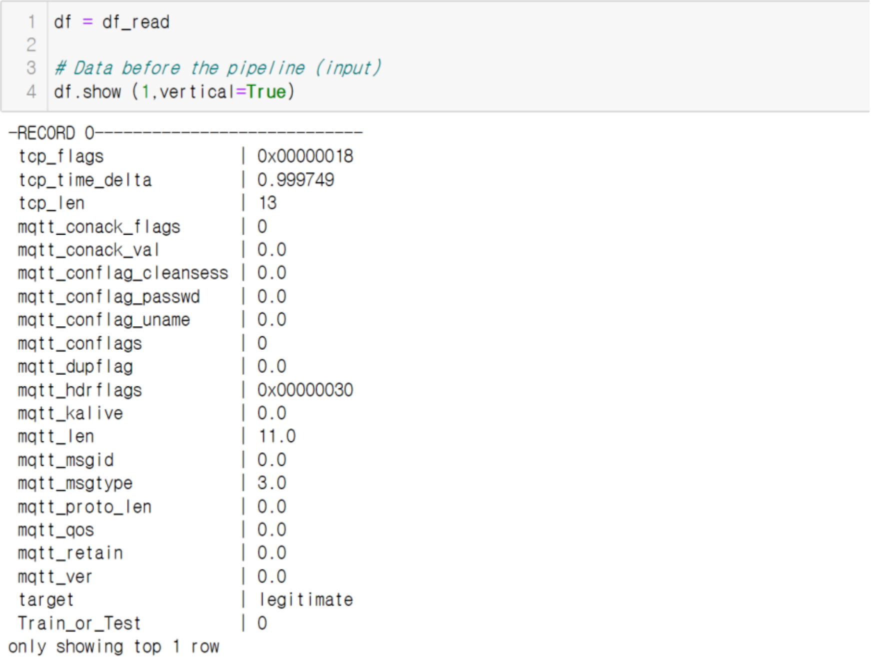
- Data frame with dropped/renamed columns (PgAdmin4)



	tcp_flags	tcp_time_delta	tcp_len	mqtt_conack_flags	mqtt_conack_val	mqtt_conflag_cleansess	mqtt_conflag_passwd	mqtt_conflag_uname	mqtt_conflags	mqtt_dupflag	mqtt_hdrflags	mqtt_k
	text	text	text	text	text	text	text	text	text	text	text	text
1	0x000000...	0.999749	13	0	0.0	0.0	0.0	0.0	0	0.0	0x00000030	0.0
2	0x000000...	0.0	0	0	0.0	0.0	0.0	0.0	0	0.0	0	0.0
3	0x000000...	4e-06	0	0	0.0	0.0	0.0	0.0	0	0.0	0	0.0
4	0x000000...	2e-06	0	0	0.0	0.0	0.0	0.0	0	0.0	0	0.0
5	0x000000...	1.5e-05	0	0	0.0	0.0	0.0	0.0	0	0.0	0	0.0
6	0x000000...	0.00023	3	0	0.0	1.0	0.0	0.0	0x00000002	0.0	0x00000010	65535.
7	0x000000...	3.6e-05	32760	0	0.0	0.0	0.0	0.0	0	0.0	0x00000030	0.0
8	0x000000...	7.9e-05	12	0	0.0	0.0	0.0	0.0	0	0.0	0x00000030	0.0
9	0x000000...	5e-06	13	0	0.0	0.0	0.0	0.0	0	0.0	0x00000030	0.0
10	0x000000...	1e-06	124	0	0.0	0.0	0.0	0.0	0	0.0	0x00000032	0.0
11	0x000000...	0.000199	28	0	0.0	1.0	1.0	1.0	0x000000c2	0.0	0x00000010	60.0
12	0x000000...	5e-06	0	0	0.0	0.0	0.0	0.0	0	0.0	0	0.0
13	0x000000...	3e-06	0	0	0.0	0.0	0.0	0.0	0	0.0	0	0.0
14	0x000000...	1.869159	32760	0	0.0	0.0	0.0	0.0	0	0.0	0x00000030	0.0
15	0x000000...	0.002058	4	0x00000000	5.0	0.0	0.0	0.0	0	0.0	0x00000020	0.0
16	0x000000...	5e-06	0	0	0.0	0.0	0.0	0.0	0	0.0	0	0.0
17	0x000000...	5e-06	0	0	0.0	0.0	0.0	0.0	0	0.0	0	0.0
18	0x000000...	0.0	0	0	0.0	0.0	0.0	0.0	0	0.0	0	0.0
19	0x000000...	0.000177	10	0	0.0	0.0	0.0	0.0	0	0.0	0x00000030	0.0
20	0x000000...	9.1e-05	13	0	0.0	0.0	0.0	0.0	0	0.0	0x00000030	0.0
21	0x000000...	0.999803	13	0	0.0	0.0	0.0	0.0	0	0.0	0x00000030	0.0

Total rows: 1000 of 999763 Query complete 00:00:01.357 Ln 1, Col 1

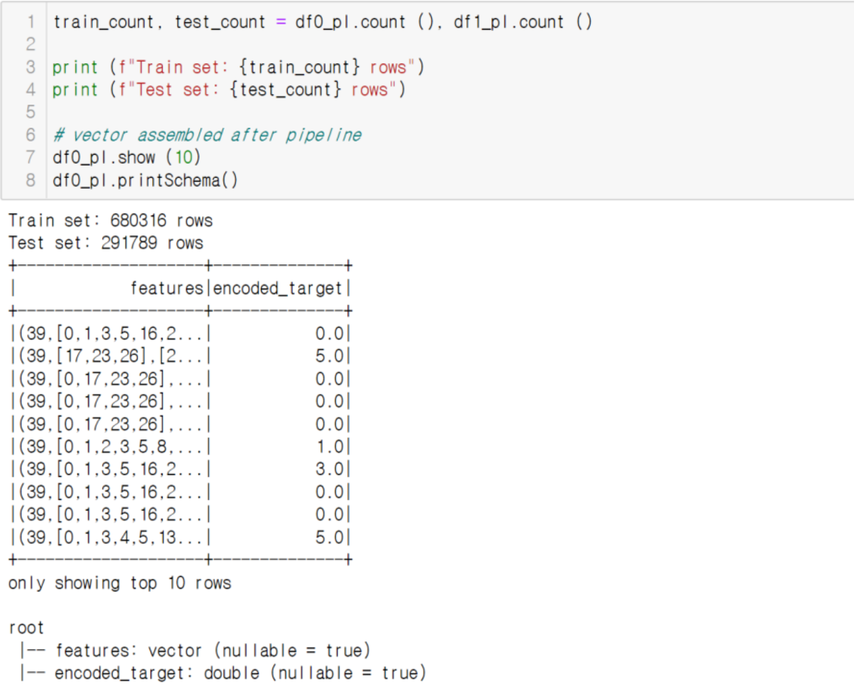
- Data frame before Pipeline



```
df = df_read

# Data before the pipeline (input)
df.show (1,vertical=True)
```

```
-RECORD 0----------------------------
 tcp_flags              | 0x00000018
 tcp_time_delta         | 0.999749
 tcp_len                | 13
 mqtt_conack_flags      | 0
 mqtt_conack_val        | 0.0
 mqtt_conflag_cleansess | 0.0
 mqtt_conflag_passwd    | 0.0
 mqtt_conflag_uname     | 0.0
 mqtt_conflags          | 0
 mqtt_dupflag           | 0.0
 mqtt_hdrflags          | 0x00000030
 mqtt_kalive            | 0.0
 mqtt_len               | 11.0
 mqtt_msgid             | 0.0
 mqtt_msgtype           | 3.0
 mqtt_proto_len         | 0.0
 mqtt_qos               | 0.0
 mqtt_retain            | 0.0
 mqtt_ver               | 0.0
 target                 | legitimate
 Train_or_Test          | 0
only showing top 1 row
```

- Data frame after Pipeline

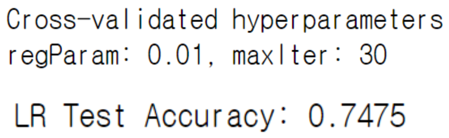


```
train_count, test_count = df0_pl.count (), df1_pl.count ()

print (f"Train set: {train_count} rows")
print (f"Test set: {test_count} rows")

# vector assembled after pipeline
df0_pl.show (10)
df0_pl.printSchema()
```

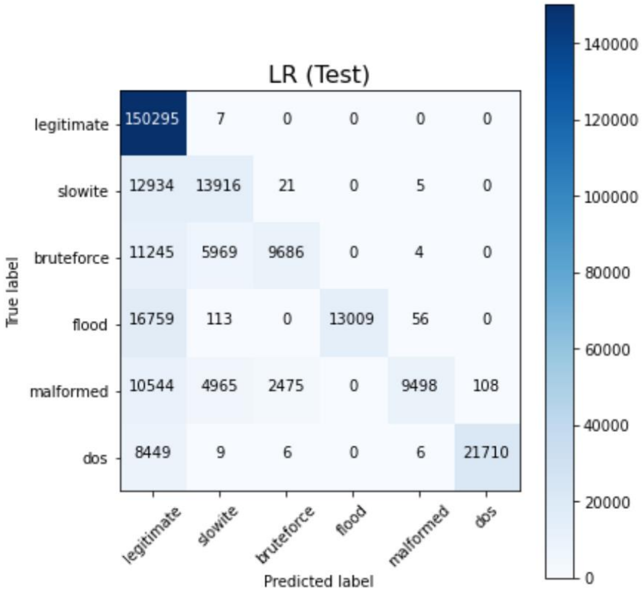
```
Train set: 680316 rows
Test set: 291789 rows
+--------------------+--------------+
|            features|encoded_target|
+--------------------+--------------+
|(39,[0,1,3,5,16,2...|           0.0|
|(39,[17,23,26],[2...|           5.0|
|(39,[0,17,23,26],...|           0.0|
|(39,[0,17,23,26],...|           0.0|
|(39,[0,17,23,26],...|           0.0|
|(39,[0,1,2,3,5,8,...|           1.0|
|(39,[0,1,3,5,16,2...|           3.0|
|(39,[0,1,3,5,16,2...|           0.0|
|(39,[0,1,3,5,16,2...|           0.0|
|(39,[0,1,3,4,5,13...|           5.0|
+--------------------+--------------+
only showing top 10 rows

root
 |-- features: vector (nullable = true)
 |-- encoded_target: double (nullable = true)
```

PySpark ML Results

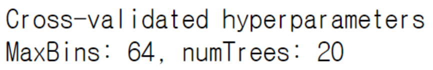
- Logistic Regression



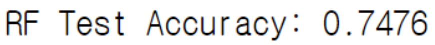
```
Cross-validated hyperparameters
regParam: 0.01, maxIter: 30

LR Test Accuracy: 0.7475
```



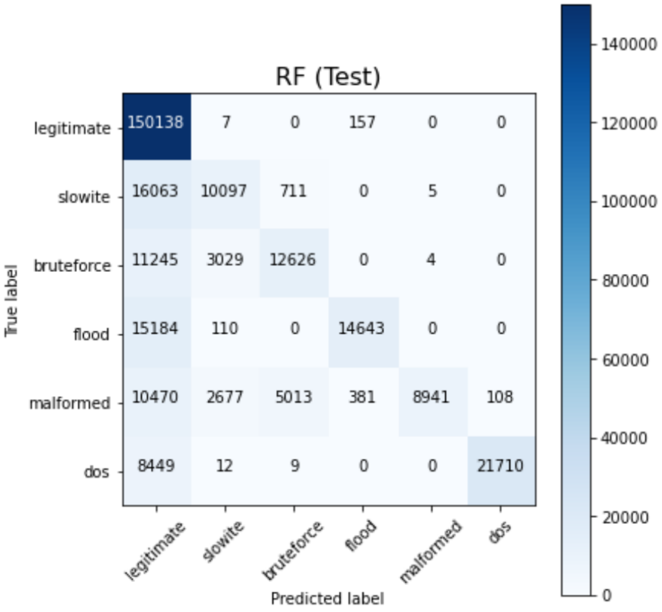
- Random Forest Classifier



```
Cross-validated hyperparameters
MaxBins: 64, numTrees: 20
```



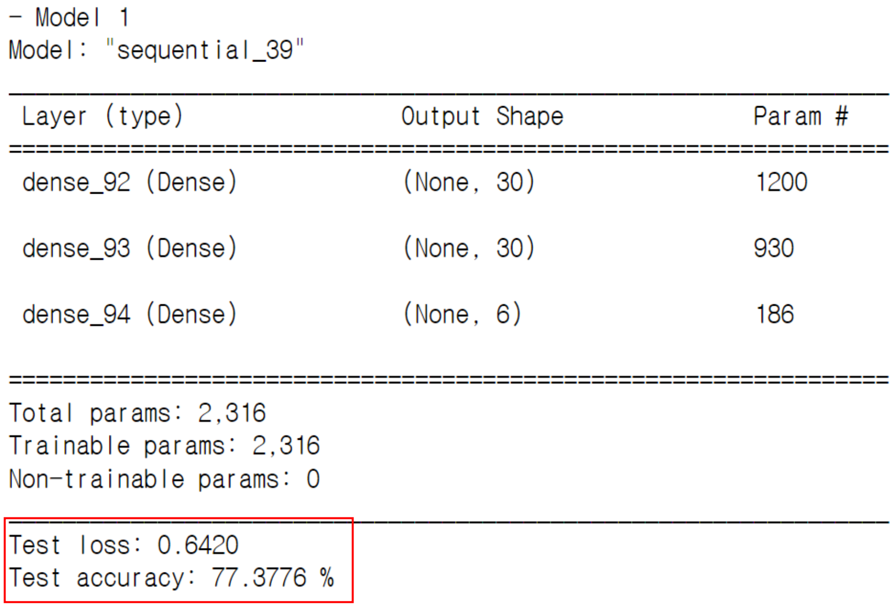
```
RF Test Accuracy: 0.7476
```



True label \ Predicted label	legitimate	slowite	bruteforce	flood	malformed	dos
legitimate	150138	7	0	157	0	0
slowite	16063	10097	711	0	5	0
bruteforce	11245	3029	12626	0	4	0
flood	15184	110	0	14643	0	0
malformed	10470	2677	5013	381	8941	108
dos	8449	12	9	0	0	21710

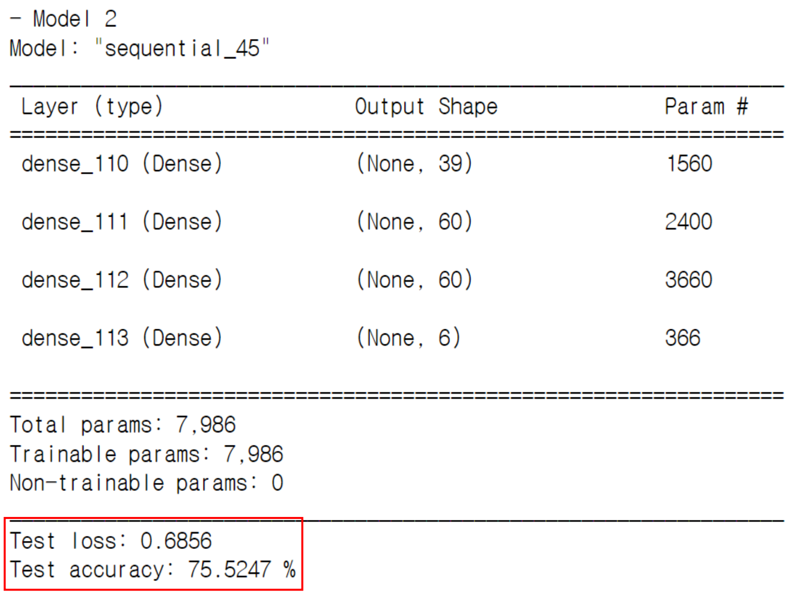
Tensorflow ML Results

- Model 1 (cross-validated: depth=2, width=30, optimizer=Adam)



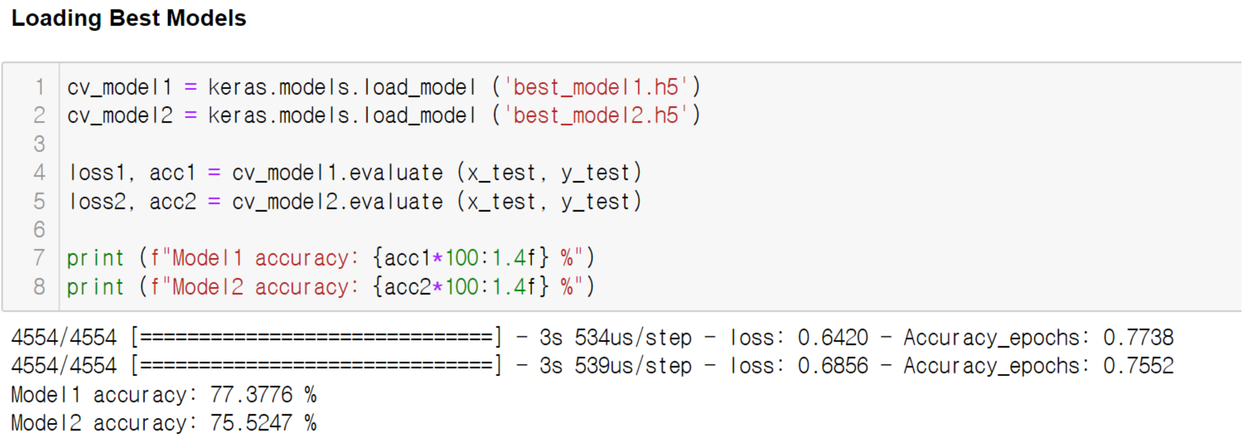
```
- Model 1
Model: "sequential_39"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 dense_92 (Dense)            (None, 30)                1200

 dense_93 (Dense)            (None, 30)                930

 dense_94 (Dense)            (None, 6)                 186

=================================================================
Total params: 2,316
Trainable params: 2,316
Non-trainable params: 0
_________________________________________________________________
Test loss: 0.6420
Test accuracy: 77.3776 %
```

- Model 2 (cross-validated: learning rate = 0.01, activation function=ReLU)



```
- Model 2
Model: "sequential_45"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 dense_110 (Dense)           (None, 39)                1560

 dense_111 (Dense)           (None, 60)                2400

 dense_112 (Dense)           (None, 60)                3660

 dense_113 (Dense)           (None, 6)                 366

=================================================================
Total params: 7,986
Trainable params: 7,986
Non-trainable params: 0
_________________________________________________________________
Test loss: 0.6856
Test accuracy: 75.5247 %
```

- Saving/Loading best models using Keras



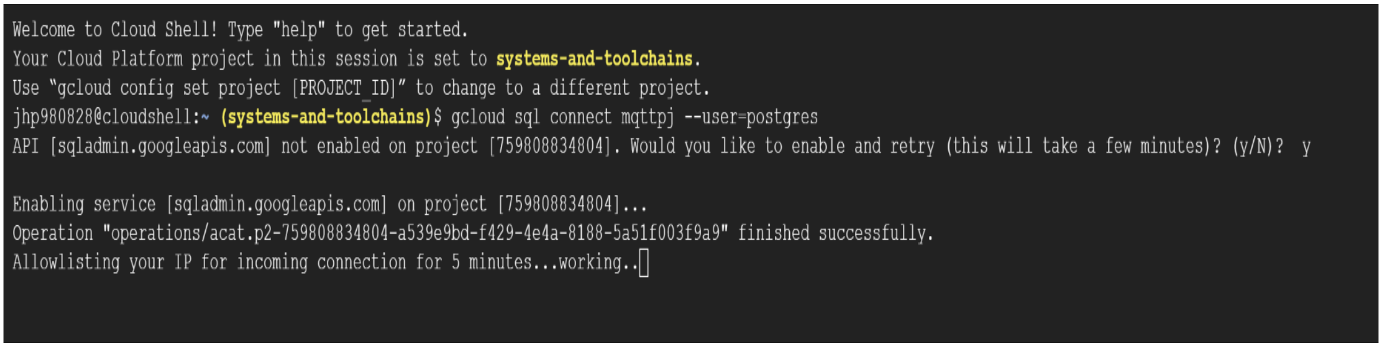
Loading Best Models

```
cv_model1 = keras.models.load_model ('best_model1.h5')
cv_model2 = keras.models.load_model ('best_model2.h5')

loss1, acc1 = cv_model1.evaluate (x_test, y_test)
loss2, acc2 = cv_model2.evaluate (x_test, y_test)

print (f"Model1 accuracy: {acc1*100:1.4f} %")
print (f"Model2 accuracy: {acc2*100:1.4f} %")
```

```
4554/4554 [==============================] - 3s 534us/step - loss: 0.6420 - Accuracy_epochs: 0.7738
4554/4554 [==============================] - 3s 539us/step - loss: 0.6856 - Accuracy_epochs: 0.7552
Model1 accuracy: 77.3776 %
Model2 accuracy: 75.5247 %
```

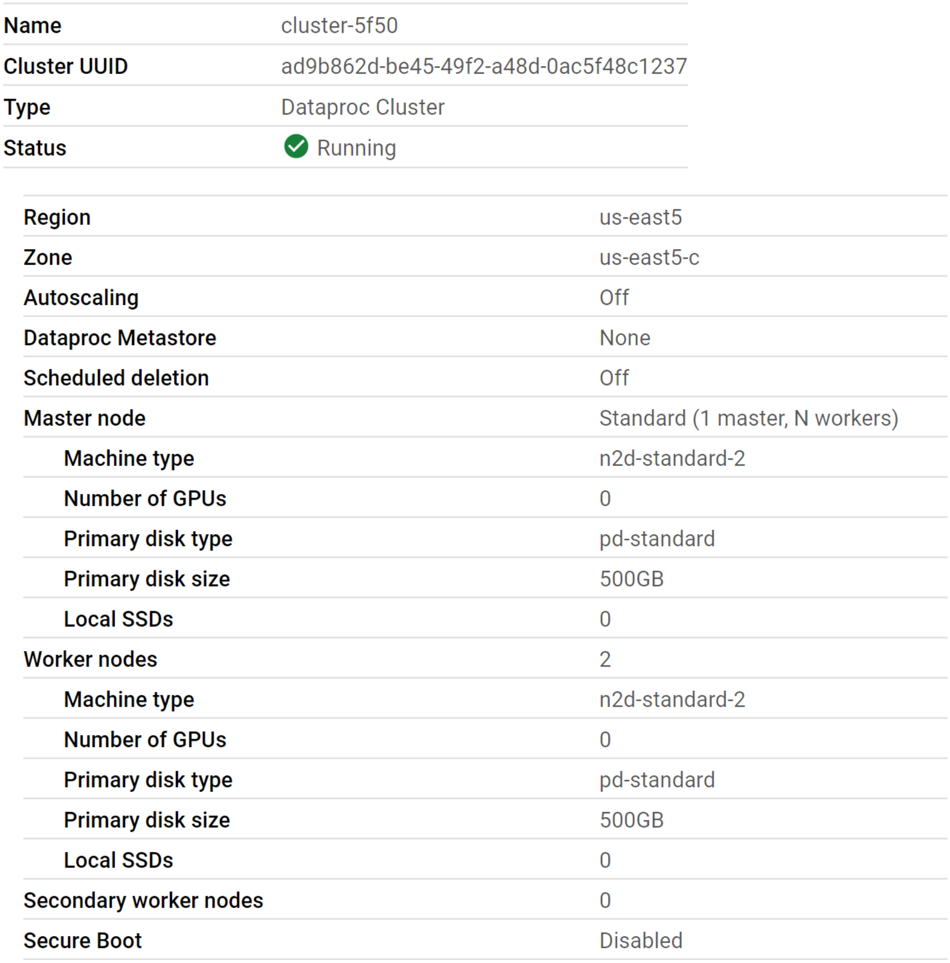
TASK IV

Connecting to Google Cloud SQL



```
Welcome to Cloud Shell! Type "help" to get started.
Your Cloud Platform project in this session is set to systems-and-toolchains.
Use "gcloud config set project [PROJECT_ID]" to change to a different project.
jhp980828@cloudshell:~ (systems-and-toolchains)$ gcloud sql connect mqttpj --user=postgres
API [sqladmin.googleapis.com] not enabled on project [759808834804]. Would you like to enable and retry (this will take a few minutes)? (y/N)?  y

Enabling service [sqladmin.googleapis.com] on project [759808834804]...
Operation "operations/acat.p2-759808834804-a539e9bd-f429-4e4a-8188-5a51f003f9a9" finished successfully.
Allowlisting your IP for incoming connection for 5 minutes...working..
```

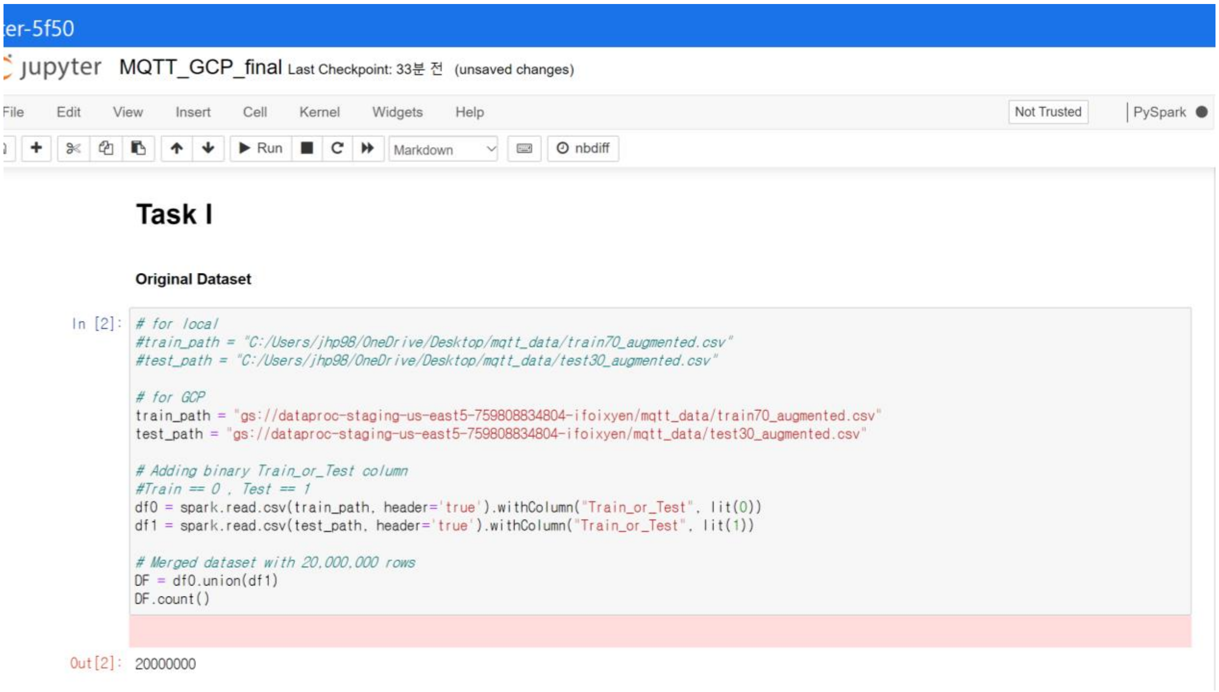
Cluster details



Name	cluster-5f50
Cluster UUID	ad9b862d-be45-49f2-a48d-0ac5f48c1237
Type	Dataproc Cluster
Status	✓ Running

Region	us-east5
Zone	us-east5-c
Autoscaling	Off
Dataproc Metastore	None
Scheduled deletion	Off
Master node	Standard (1 master, N workers)
Machine type	n2d-standard-2
Number of GPUs	0
Primary disk type	pd-standard
Primary disk size	500GB
Local SSDs	0
Worker nodes	2
Machine type	n2d-standard-2
Number of GPUs	0
Primary disk type	pd-standard
Primary disk size	500GB
Local SSDs	0
Secondary worker nodes	0
Secure Boot	Disabled

Example Run

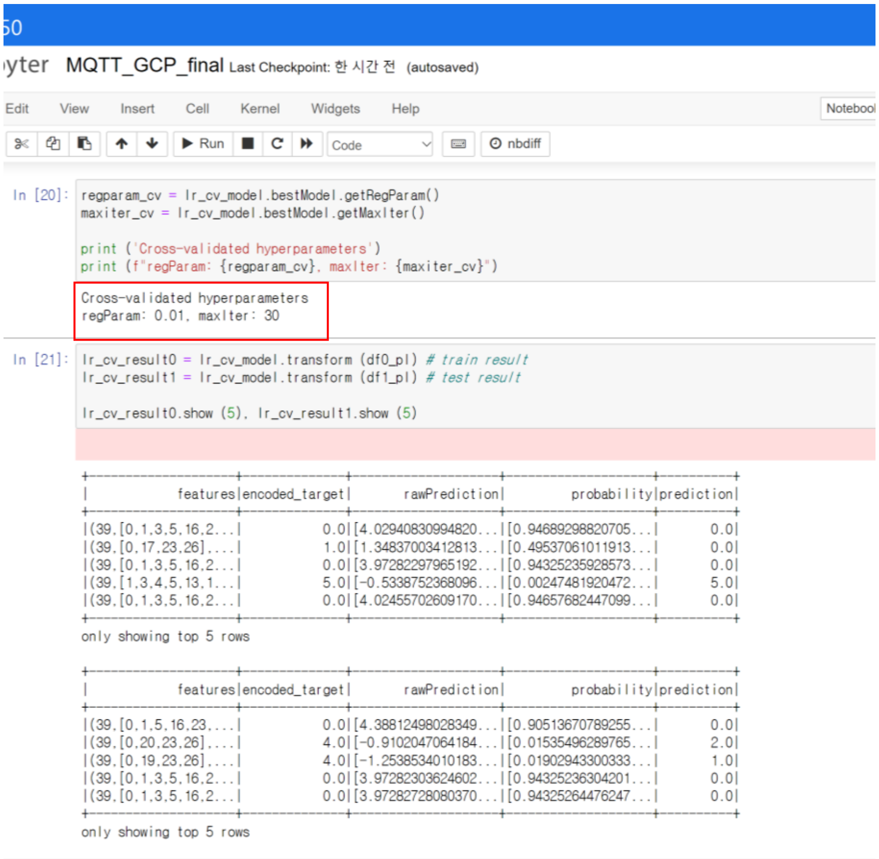


Task I

Original Dataset

```
In [2]:
# for local
#train_path = "C:/Users/jhp98/OneDrive/Desktop/mqtt_data/train70_augmented.csv"
#test_path = "C:/Users/jhp98/OneDrive/Desktop/mqtt_data/test30_augmented.csv"

# for GCP
train_path = "gs://dataproc-staging-us-east5-759808834804-ifoixyen/mqtt_data/train70_augmented.csv"
test_path = "gs://dataproc-staging-us-east5-759808834804-ifoixyen/mqtt_data/test30_augmented.csv"

# Adding binary Train_or_Test column
#Train == 0 , Test == 1
df0 = spark.read.csv(train_path, header='true').withColumn("Train_or_Test", lit(0))
df1 = spark.read.csv(test_path, header='true').withColumn("Train_or_Test", lit(1))

# Merged dataset with 20,000,000 rows
DF = df0.union(df1)
DF.count()

Out[2]: 20000000
```

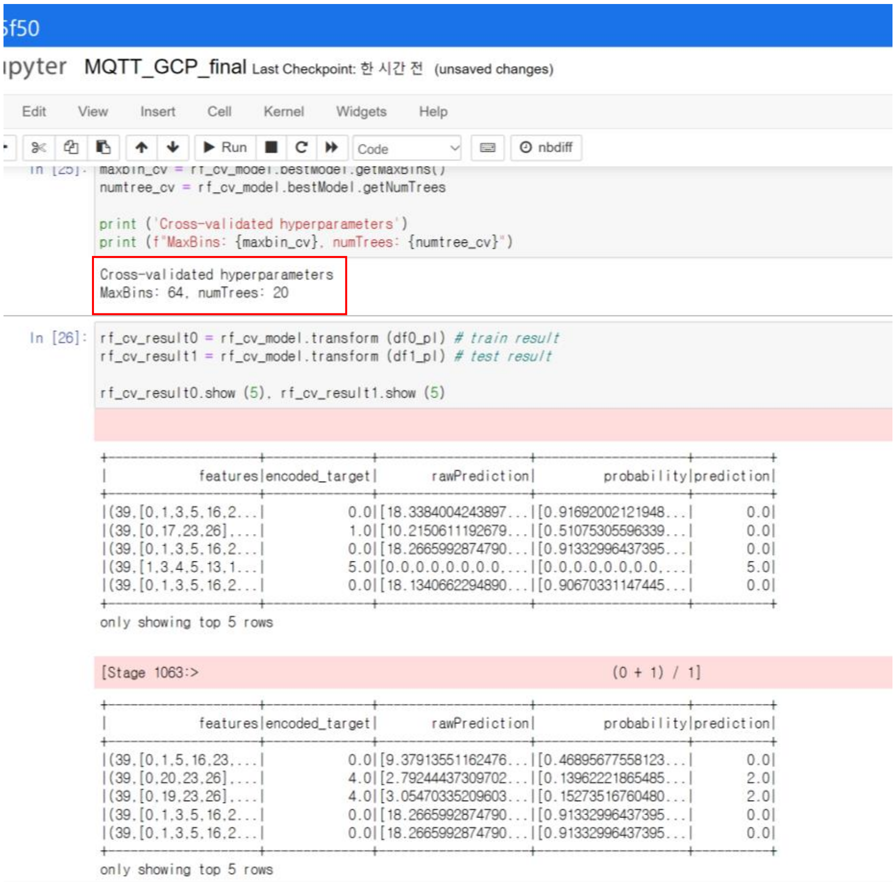
- Logistic Regression Result



```
In [20]:
regparam_cv = lr_cv_model.bestModel.getRegParam()
maxiter_cv = lr_cv_model.bestModel.getMaxIter()

print ('Cross-validated hyperparameters')
print (f"regParam: {regparam_cv}, maxIter: {maxiter_cv}")

Cross-validated hyperparameters
regParam: 0.01, maxIter: 30
```

```
In [21]:
lr_cv_result0 = lr_cv_model.transform (df0_pl) # train result
lr_cv_result1 = lr_cv_model.transform (df1_pl) # test result

lr_cv_result0.show (5), lr_cv_result1.show (5)

+--------------------+--------------+--------------------+--------------------+----------+
|            features|encoded_target|       rawPrediction|         probability|prediction|
+--------------------+--------------+--------------------+--------------------+----------+
|(39,[0,1,3,5,16,2...|           0.0|[4.02940830994820...|[0.94689298820705...|       0.0|
|(39,[0,17,23,26],...|           1.0|[1.34837003412813...|[0.49537061011913...|       0.0|
|(39,[0,1,3,5,16,2...|           0.0|[3.97282297965192...|[0.94325235928573...|       0.0|
|(39,[1,3,4,5,13,1...|           5.0|[-0.5338752368096...|[0.00247481920472...|       5.0|
|(39,[0,1,3,5,16,2...|           0.0|[4.02455702609170...|[0.94657682447099...|       0.0|
+--------------------+--------------+--------------------+--------------------+----------+
only showing top 5 rows

+--------------------+--------------+--------------------+--------------------+----------+
|            features|encoded_target|       rawPrediction|         probability|prediction|
+--------------------+--------------+--------------------+--------------------+----------+
|(39,[0,1,5,16,23,...|           0.0|[4.38812498028349...|[0.90513670789255...|       0.0|
|(39,[0,20,23,26],...|           4.0|[-0.9102047064184...|[0.01535496289765...|       2.0|
|(39,[0,19,23,26],...|           4.0|[-1.2538534010183...|[0.01902943300333...|       1.0|
|(39,[0,1,3,5,16,2...|           0.0|[3.97282303624602...|[0.94325236304201...|       0.0|
|(39,[0,1,3,5,16,2...|           0.0|[3.97282728080370...|[0.94325264476247...|       0.0|
+--------------------+--------------+--------------------+--------------------+----------+
only showing top 5 rows
```

- Random Forest Result

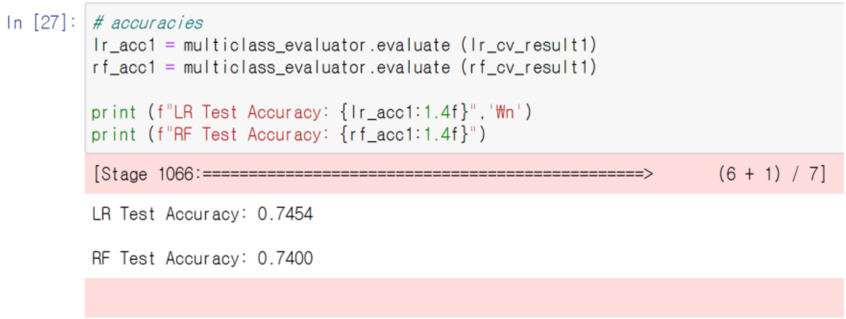

f50
ıpyter MQTT_GCP_final Last Checkpoint: 한 시간 전 (unsaved changes)
Edit
View
Insert
Cell
Kernel
Widgets
Help
Run
Code
nbdiff

```
In [25]: maxbin_cv = rf_cv_model.bestModel.getMaxBins()
         numtree_cv = rf_cv_model.bestModel.getNumTrees

         print ('Cross-validated hyperparameters')
         print (f"MaxBins: {maxbin_cv}, numTrees: {numtree_cv}")

         Cross-validated hyperparameters
         MaxBins: 64, numTrees: 20
```

```
In [26]: rf_cv_result0 = rf_cv_model.transform (df0_pl) # train result
         rf_cv_result1 = rf_cv_model.transform (df1_pl) # test result

         rf_cv_result0.show (5), rf_cv_result1.show (5)
```

```
+--------------------+--------------+--------------------+--------------------+----------+
|            features|encoded_target|       rawPrediction|         probability|prediction|
+--------------------+--------------+--------------------+--------------------+----------+
|(39,[0,1,3,5,16,2...|           0.0|[18.3384004243897...|[0.91692002121948...|       0.0|
|(39,[0,17,23,26],...|           1.0|[10.2150611192679...|[0.51075305596339...|       0.0|
|(39,[0,1,3,5,16,2...|           0.0|[18.2665992874790...|[0.91332996437395...|       0.0|
|(39,[1,3,4,5,13,1...|           5.0|[0.0,0.0,0.0,0.0,...|[0.0,0.0,0.0,0.0,...|       5.0|
|(39,[0,1,3,5,16,2...|           0.0|[18.1340662294890...|[0.90670331147445...|       0.0|
+--------------------+--------------+--------------------+--------------------+----------+
only showing top 5 rows

[Stage 1063:>                                                       (0 + 1) / 1]

+--------------------+--------------+--------------------+--------------------+----------+
|            features|encoded_target|       rawPrediction|         probability|prediction|
+--------------------+--------------+--------------------+--------------------+----------+
|(39,[0,1,5,16,23,...|           0.0|[9.37913551162476...|[0.46895677558123...|       0.0|
|(39,[0,20,23,26],...|           4.0|[2.79244437309702...|[0.13962221865485...|       2.0|
|(39,[0,19,23,26],...|           4.0|[3.05470335209603...|[0.15273516760480...|       2.0|
|(39,[0,1,3,5,16,2...|           0.0|[18.2665992874790...|[0.91332996437395...|       0.0|
|(39,[0,1,3,5,16,2...|           0.0|[18.2665992874790...|[0.91332996437395...|       0.0|
+--------------------+--------------+--------------------+--------------------+----------+
only showing top 5 rows
```

- Test Accuracy



```
In [27]: # accuracies
         lr_acc1 = multiclass_evaluator.evaluate (lr_cv_result1)
         rf_acc1 = multiclass_evaluator.evaluate (rf_cv_result1)

         print (f"LR Test Accuracy: {lr_acc1:1.4f}",'₩n')
         print (f"RF Test Accuracy: {rf_acc1:1.4f}")

         [Stage 1066:================================================>        (6 + 1) / 7]

         LR Test Accuracy: 0.7454

         RF Test Accuracy: 0.7400
```