databricks Assignment 3_Jessie Ma

(https://databricks.com)

How to Process IoT Device JSON Data Using Dataset

Reading JSON as a Dataset

Use the Scala case class *DeviceIoTData* to convert the JSON device data into a dataframe. There is GeoIP information for each device entry:

- IP address
- ISO-3166-1 two and three letter codes
- Country Name
- Latitude and longitude

With these attributes as part of the device data, we can map and visualize them as needed. For each IP associated with a *device_id*, I optained the above attributes from a webservice at http://freegeoip.net/csv/ip (http://freegeoip.net/csv/ip)

{"device_id": 198164, "device_name": "sensor-pad-198164owomcJZ", "ip": "80.55.20.25", "cca2": "PL", "cca3": "POL", "cn": "Poland", "latitude": 53.080000, "longitude": 18.620000, "scale": "Celsius", "temp": 21, "humidity": 65, "battery_level": 8, "c02_level": 1408, "lcd": "red", "timestamp" :1458081226051 }

This dataset is avaialbe from Public S3 bucket //databricks-public-datasets/data/iot or https://github.com/dmatrix/examples/blob/master/spark/databricks/notebooks/py/data/iot_devices.json (https://github.com/dmatrix/examples/blob/master/spark/databricks/notebooks/py/data/iot_devices.json)

```
%fs ls "FileStore/tables/"
```

Show cell

```
# read the json file and create the dataframe


file_location = "/FileStore/tables/iot_devices.json"
file_type = "json"

# CSV options
infer_schema = "false"
first_row_is_header = "false"
delimiter = ","

# The applied options are for CSV files. For other file types, these will be ignored.
df = spark.read.format(file_type) \
  .option("inferSchema", infer_schema) \
  .option("header", first_row_is_header) \
  .option("sep", delimiter) \
  .load(file_location)

display(df)
```

Table

	battery_level	c02_level	cca2	cca3	cn	device_id	device_name
1	8	868	US	USA	United States	1	meter-gauge-1xbYRYcj
2	7	1473	NO	NOR	Norway	2	sensor-pad-2n2Pea

2	7	1473	NO	NOR	Norway	2	sensor-pad-2n2Pea
3	2	1556	IT	ITA	Italy	3	device-mac-36TWSKiT
4	6	1080	US	USA	United States	4	sensor-pad-4mzWkz
5	4	931	PH	PHL	Philippines	5	therm-stick-5gimpUrBB
6	3	1210	US	USA	United States	6	sensor-pad-6al7RTAobR
7	3	1129	CN	CHN	China	7	meter-gauge-7GeDoanM

10,000 rows | Truncated data

```
# Create a view or table

temp_table_name = "iot_devices_json"

df.createOrReplaceTempView(temp_table_name)
```

```
df.count()
```

```
Out[3]: 198164
```

Displaying your Dataset

```
display(df)
```

Table

	battery_level	c02_level	cca2	cca3	cn	device_id	device_name
1	8	868	US	USA	United States	1	meter-gauge-1xbYRYcj
2	7	1473	NO	NOR	Norway	2	sensor-pad-2n2Pea
3	2	1556	IT	ITA	Italy	3	device-mac-36TWSKiT
4	6	1080	US	USA	United States	4	sensor-pad-4mzWkz
5	4	931	PH	PHL	Philippines	5	therm-stick-5gimpUrBB
6	3	1210	US	USA	United States	6	sensor-pad-6al7RTAobR
7	3	1129	CN	CHN	China	7	meter-gauge-7GeDoanM

10,000 rows | Truncated data

Data exploration

Top five entries

```
df.take(5)
```

```
Out[5]: [Row(battery_level=8, c02_level=868, cca2='US', cca3='USA', cn='United States', device_id=1, device_name='meter-gauge-1xbYRY
cj', humidity=51, ip='68.161.225.1', latitude=38.0, lcd='green', longitude=-97.0, scale='Celsius', temp=34, timestamp=145844405409
3),
 Row(battery_level=7, c02_level=1473, cca2='NO', cca3='NOR', cn='Norway', device_id=2, device_name='sensor-pad-2n2Pea', humidity=70,
ip='213.161.254.1', latitude=62.47, lcd='red', longitude=6.15, scale='Celsius', temp=11, timestamp=1458444054119),
 Row(battery_level=2, c02_level=1556, cca2='IT', cca3='ITA', cn='Italy', device_id=3, device_name='device-mac-36TWSKiT', humidity=4
4, ip='88.36.5.1', latitude=42.83, lcd='red', longitude=12.83, scale='Celsius', temp=19, timestamp=1458444054120),
 Row(battery_level=6, c02_level=1080, cca2='US', cca3='USA', cn='United States', device_id=4, device_name='sensor-pad-4mzWkz', humid
ity=32, ip='66.39.173.154', latitude=44.06, lcd='yellow', longitude=-121.32, scale='Celsius', temp=28, timestamp=1458444054121),
 Row(battery_level=4, c02_level=931, cca2='PH', cca3='PHL', cn='Philippines', device_id=5, device_name='therm-stick-5gimpUrBB', humi
dity=62, ip='203.82.41.9', latitude=14.58, lcd='green', longitude=120.97, scale='Celsius', temp=25, timestamp=1458444054122)]
```

For all relational expressions, the Catalyst Optimizer (https://databricks.com/blog/2015/04/13/deep-dive-into-spark-sqls-catalyst-optimizer.html) will formulate an optimized logical and physical plan for execution, and Tungsten (https://databricks.com/blog/2015/04/28/project-tungsten-bringing-spark-closer-to-bare-metal.html) engine will optimize the

generated code. For our *DeviceIoTData*, it will use its standard encoders to optimize its binary internal representation, hence decrease the size of generated code, minimize the bytes transfered over the networks between nodes, and execute faster.

```
# issue select, map, filter operations on the dataframes

from pyspark.sql.functions import col, asc
TempFilter = df.filter(col("temp") > 30).filter(col("humidity") > 70)
display(TempFilter)
```

Table

	battery_level	c02_level	cca2	cca3	cn	device_id	device_name
1	0	1466	US	USA	United States	17	meter-gauge-17zb8Fghhl
2	9	986	FR	FRA	France	48	sensor-pad-48jt4eL
3	8	1436	US	USA	United States	54	sensor-pad-5410CWPrNb6
4	4	1090	US	USA	United States	63	device-mac-63GL4xSaZbj
5	4	1072	PH	PHL	Philippines	81	device-mac-81nsKomrRe
6	3	1076	FR	FRA	France	82	sensor-pad-82HJm6yP
7	9	1221	DE	DEU	Germany	83	meter-gauge-83lLWufdrzWE

10,000 rows | Truncated data

Use filter to filter out dataframe rows that met the temperature and humidity predicate

```
# filter out rows that meet the temperature and humimdity predicate
TempFilter10 = df.filter(col("temp") > 30).filter(col("humidity") > 70).take(10)
TempFilter10
```

```
Out[7]: [Row(battery_level=0, c02_level=1466, cca2='US', cca3='USA', cn='United States', device_id=17, device_name='meter-gauge-17zb8Fghhl', humidity=98, ip='161.188.212.254', latitude=39.95, lcd='red', longitude=-75.16, scale='Celsius', temp=31, timestamp=1458444054129),
 Row(battery_level=9, c02_level=986, cca2='FR', cca3='FRA', cn='France', device_id=48, device_name='sensor-pad-48jt4eL', humidity=97, ip='90.37.208.1', latitude=43.88, lcd='green', longitude=4.9, scale='Celsius', temp=31, timestamp=1458444054151),
 Row(battery_level=8, c02_level=1436, cca2='US', cca3='USA', cn='United States', device_id=54, device_name='sensor-pad-5410CWPrNb6', humidity=73, ip='204.15.64.249', latitude=32.89, lcd='red', longitude=-117.13, scale='Celsius', temp=34, timestamp=1458444054155),
 Row(battery_level=4, c02_level=1090, cca2='US', cca3='USA', cn='United States', device_id=63, device_name='device-mac-63GL4xSaZbj', humidity=91, ip='66.198.198.1', latitude=44.56, lcd='yellow', longitude=-105.67, scale='Celsius', temp=31, timestamp=1458444054162),
 Row(battery_level=4, c02_level=1072, cca2='PH', cca3='PHL', cn='Philippines', device_id=81, device_name='device-mac-81nsKomrRe', humidity=90, ip='222.127.71.1', latitude=14.55, lcd='yellow', longitude=121.04, scale='Celsius', temp=31, timestamp=1458444054172),
 Row(battery_level=3, c02_level=1076, cca2='FR', cca3='FRA', cn='France', device_id=82, device_name='sensor-pad-82HJm6yP', humidity=76, ip='213.162.50.33', latitude=48.86, lcd='yellow', longitude=2.35, scale='Celsius', temp=32, timestamp=1458444054172),
 Row(battery_level=9, c02_level=1221, cca2='DE', cca3='DEU', cn='Germany', device_id=83, device_name='meter-gauge-83lLWufdrzWE', humidity=96, ip='62.214.32.222', latitude=51.0, lcd='yellow', longitude=9.0, scale='Celsius', temp=31, timestamp=1458444054173),
 Row(battery_level=2, c02_level=1182, cca2='US', cca3='USA', cn='United States', device_id=108, device_name='sensor-pad-108NG6gl2jPi', humidity=82, ip='208.35.184.254', latitude=34.2, lcd='yellow', longitude=-118.82, scale='Celsius', temp=34, timestamp=1458444054187),
```

```
# Mapping four fields- temp, device_name, device_id, cca3
dfTempMap = df.where((col("temp") > 25)).rdd.map(lambda d: (d.temp, d.device_name, d.device_id, d.cca3))
dfTempMap
```

```
Out[8]: PythonRDD[28] at RDD at PythonRDD.scala:58
```

```
display(dfTempMap.toDF())
```

Table

	_1	_2	_3	_4
1	34	meter-gauge-1xbYRYcj	1	USA
2	28	sensor-pad-4mzWkz	4	USA

3	27	sensor-pad-6al7RTAobR	6	USA
4	27	sensor-pad-8xUD6pzsQI	8	JPN
5	26	sensor-pad-10BsywSYUF	10	USA
6	31	meter-gauge-17zb8Fghhl	17	USA
7	31	sensor-pad-18XULN9Xv	18	CHN

10,000 rows | Truncated data

Now use the filter() method that is equivalent as the where() method used above.

```
dfTemp25 = df.filter(col("temp") > 25).rdd.map(lambda d: (d.temp, d.device_name, d.device_id, d.cca3))

display(dfTemp25.toDF())
```

Table

	_1	_2	_3	_4
1	34	meter-gauge-1xbYRYcj	1	USA
2	28	sensor-pad-4mzWkz	4	USA
3	27	sensor-pad-6al7RTAobR	6	USA
4	27	sensor-pad-8xUD6pzsQI	8	JPN
5	26	sensor-pad-10BsywSYUF	10	USA
6	31	meter-gauge-17zb8Fghhl	17	USA
7	31	sensor-pad-18XULN9Xv	18	CHN

10,000 rows | Truncated data

select() where battery_level is greater than 6, sort in asceding order on C02_level.

```
display(df.select("battery_level", "c02_level", "device_name").where(col("battery_level") > 6).sort(col("c02_level")))
```

Table

	battery_level	c02_level	device_name
1	8	800	sensor-pad-185138dUJqFY0
2	7	800	sensor-pad-33956IIjAk60a
3	8	800	device-mac-53529CWnXme
4	9	800	device-mac-36441uFJaQeBMv7
5	8	800	sensor-pad-83310txFDw
6	8	800	meter-gauge-37261gwzOn
7	8	800	meter-gauge-60509Mzi9YApS

10,000 rows | Truncated data

Let's see how to use groupBy() and avg(). Let's take all temperatures readings > 25, along with their corresponding devices' humidity, groupBy ccca3 country code, and compute averages. Plot the resulting Dataset.

```
from pyspark.sql.functions import avg

dfAvgTmp = df.filter(col("temp") > 25).rdd.map(lambda d: (d.temp, d.humidity, d.cca3)).toDF().groupBy("_3").agg(avg("_1"), avg("_2"))

display(dfAvgTmp)
```

Table Legacy - Line (v1)

	_3	avg(_1)	avg(_2)

1	PSE	30.88888888888889	62.22222222222222
2	POL	29.929577464788732	62.045271629778675
3	LVA	29.721804511278197	63.29323308270677
4	BRB	29.63157894736842	61.21052631578947
5	ZMB	30	60
6	BRA	30.09396551724138	61.126724137931035
7	MOZ	29.8	67.8

195 rows

Visualizing datasets

Finally, the fun bit!

Data without visualization without a narrative arc, to infer insights or to see a trend, is useless. We always desire to make sense of the results.

By saving our Dataset, as a temporary table, I can issue complex SQL queries against it and visualize the results, using notebook's myriad plotting options.

```
df.createOrReplaceTempView("iot_device_data")
```

Count all devices for a partiular country and map them

```
%sql select cca3, count(distinct device_id) as device_id from iot_device_data group by cca3 order by device_id desc limit 100
```

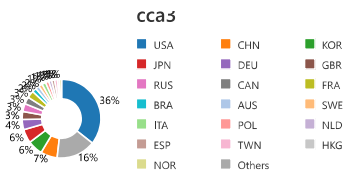
Visualization

100 rows

Let's visualize the results as a pie chart and distribution for devices in the country where C02 are high.

```
%sql select cca3, c02_level from iot_device_data where c02_level > 1400 order by c02_level desc
```

Visualization



Error plotting over all results: Show error. | Truncated to 10,000 rows.

Select all countries' devices with high-levels of C02 and group by cca3 and order by device_ids

```
%sql select cca3, count(distinct device_id) as device_id from iot_device_data where lcd == 'red' group by cca3 order by device_id desc
```

Visualization

100 rows

find out all devices in countries whose batteries need replacements

```
%sql select cca3, count(distinct device_id) as device_id from iot_device_data where battery_level == 0 group by cca3 order by device_i
```

Visualization

100 rows

Converting a Dataset to RDDs.

```
deviceEvents = df.select("device_name","cca3","c02_level").where(col("c02_level") > 1300)

eventsRDD = deviceEvents.take(10)
```

```
display(deviceEvents)
```

Table

	device_name	cca3	c02_level
1	sensor-pad-2n2Pea	NOR	1473
2	device-mac-36TWSKiT	ITA	1556
3	sensor-pad-8xUD6pzsQI	JPN	1536
4	sensor-pad-10BsywSYUF	USA	1470
5	meter-gauge-11dlMTZty	ITA	1544
6	sensor-pad-14QL93sBR0j	NOR	1346
7	sensor-pad-16aXmIJZtdO	USA	1425

10,000 rows | Truncated data

Assignment 3 Questions and Answers

1. Explain the main differences between RDDs, Dataframes and Datasets (4 marks)

The main differences between RDDs, Dataframes and Datasets are:

RDDs are used for low-level operations and control over your dataset and lack the optimization benefits offered by higher-level abstractions that Dataframes and Datasets provide. Another difference is that RDDs do not impose a schema such as a columnar format whereas Dataframes have a distributed collection of data organized into columns. Dataframes and Datasets both demonstrate higher space efficiency and performance gains than RDDs, and provide rich semantics and domain-specific APIs in addition to high-level abstractions. As for aggregations, averages and sums, RDDs are slow to perform even simple ones and grouping operations whereas both Dataframes and Datasets can perform fast aggregation on numerous large datasets. To achieve a higher degree of type-safety at compile time, utilize typed JVM objects, Catalyst optimization and Tungsten's efficient code generation, Dataset would be the API to use.

2. Answer the following questions:

2.1 How many sensor pads are reported to be from Poland (2 marks)

```
# Exploratory analysis
display(df)
df.printSchema()
```

Table

	battery_level	c02_level	cca2	cca3	cn	device_id	device_name
1	8	868	US	USA	United States	1	meter-gauge-1xbYRYcj
2	7	1473	NO	NOR	Norway	2	sensor-pad-2n2Pea
3	2	1556	IT	ITA	Italy	3	device-mac-36TWSKiT
4	6	1080	US	USA	United States	4	sensor-pad-4mzWkz
5	4	931	PH	PHL	Philippines	5	therm-stick-5gimpUrBB
6	3	1210	US	USA	United States	6	sensor-pad-6al7RTAobR
7	3	1129	CN	CHN	China	7	meter-gauge-7GeDoanM

10,000 rows | Truncated data

```
root
 |-- battery_level: long (nullable = true)
 |-- c02_level: long (nullable = true)
 |-- cca2: string (nullable = true)
 |-- cca3: string (nullable = true)
 |-- cn: string (nullable = true)
```

```
 |-- device_id: long (nullable = true)
 |-- device_name: string (nullable = true)
 |-- humidity: long (nullable = true)
 |-- ip: string (nullable = true)
 |-- latitude: double (nullable = true)
 |-- lcd: string (nullable = true)
 |-- longitude: double (nullable = true)
 |-- scale: string (nullable = true)
 |-- temp: long (nullable = true)
 |-- timestamp: long (nullable = true)
```

```
df_poland=df.filter((df.cn == 'Poland') & (df.device_name.contains('sensor-pad')))
df_poland.show()
df_poland.count()
```

```
+-------------+---------+----+----+------+---------+--------------------+--------+--------------+--------+------+---------+----
---+----+-------------+
|battery_level|c02_level|cca2|cca3|    cn|device_id|         device_name|humidity|            ip|latitude|   lcd|longitude|  sc
ale|temp|    timestamp|
+-------------+---------+----+----+------+---------+--------------------+--------+--------------+--------+------+---------+----
---+----+-------------+
|            7|     1036|  PL| POL|Poland|      170|sensor-pad-1703fywiW|      63|  212.87.11.82|   52.25|yellow|     21.0|Cels
ius|  32|1458444054223|
|            0|     1572|  PL| POL|Poland|      378|sensor-pad-378Sgq...|      71| 83.238.122.21|   52.23|   red|    21.02|Cels
ius|  15|1458444054306|
|            3|     1067|  PL| POL|Poland|      566|sensor-pad-566TPT...|      31|  149.156.8.73|   50.08|yellow|    19.92|Cels
ius|  20|1458444054360|
|            8|     1556|  PL| POL|Poland|      790|sensor-pad-790HqV...|      61|   79.139.16.1|   50.26|   red|    19.03|Cels
ius|  24|1458444054409|
|            2|      835|  PL| POL|Poland|      794|sensor-pad-7944KF...|      66|83.142.138.254|   52.23| green|    21.02|Cels
ius|  31|1458444054409|
|            4|      975|  PL| POL|Poland|      892| sensor-pad-892TMLKM|      45|  86.111.204.1|   52.23| green|    21.02|Cels
ius|  24|1458444054427|
|            9|     1462|  PL| POL|Poland|     1220|sensor-pad-1220Fr...|      46| 87.204.248.97|   52.23|   red|    21.02|Cels
ius|  17|1458444054477|
|            6|     1074|  PL| POL|Poland|     1520|sensor-pad-1520qJ...|      64|62.121.129.133|   52.23|yellow|    21.02|Cels
```

1413 sensor pads are reported to be from Poland.

2.2 How many different LCDs (distinct colors) are present in the dataset (2 marks)

```
df.select('lcd').distinct().show()
df_colors=df.groupBy("lcd").count()
df_colors.show()
```

```
+------+
|   lcd|
+------+
| green|
|yellow|
|   red|
+------+

+------+-----+
|   lcd|count|
+------+-----+
| green|49699|
|yellow|99051|
|   red|49414|
+------+-----+
```

There are 3 distinct LCD colours in the dataset: green, yellow and red. There are 49699 sensor pads with green LCDs, 99051 with yellow LCDs and 49414 with red LCDs.

2.3 Find 5 countries that have the largest number of MAC devices used (2 marks)

```
df_mac=df.filter(df.device_name.contains('mac'))
df_mac.show()
df_mac_cn=df_mac.groupBy("cn").count()
df_mac_cn.sort(col("count").desc()).show(5)
```

```
+-------------+---------+----+----+-----------------+---------+--------------------+--------+---------------+--------+------+--
-------+-------+----+-------------+
|battery_level|c02_level|cca2|cca3|               cn|device_id|         device_name|humidity|             ip|latitude|   lcd|lo
ngitude|  scale|temp|    timestamp|
+-------------+---------+----+----+-----------------+---------+--------------------+--------+---------------+--------+------+--
-------+-------+----+-------------+
|            2|     1556|  IT| ITA|            Italy|        3| device-mac-36TWSKiT|      44|      88.36.5.1|   42.83|   red|
12.83|Celsius|  19|1458444054120|
|            3|      807|  JP| JPN|            Japan|        9| device-mac-9GcjZ2pw|      85|  118.23.68.227|   35.69| green|
139.69|Celsius|  13|1458444054124|
|            9|     1259|  US| USA|    United States|       15|  device-mac-15se6mZ|      70|    67.185.72.1|   47.41|yellow|
-122.0|Celsius|  13|1458444054128|
|            5|      939|  AT| AUT|          Austria|       21|  device-mac-21sjz5h|      44|193.200.142.254|    48.2| green|
16.37|Celsius|  30|1458444054131|
|            5|     1597|  KR| KOR|Republic of Korea|       27|  device-mac-27P5wf2|      73|  218.239.168.1|   37.57|   red|
126.98|Celsius|  10|1458444054135|
|            3|      835|  RU| RUS|           Russia|       33|device-mac-33B94G...|      30| 178.23.147.134|   55.75| green|
37.62|Celsius|  15|1458444054139|
|            2|      908|  CN| CHN|            China|       39|device-mac-39iklY...|      84|     218.7.15.1|   45.75| green|
126.65|Celsius|  17|1458444054144|
|            4|      985|  IT| ITA|            Italy|       45|device-mac-45fN2C...|      74| 213.140.20.230|   42.83| green|
```

The 5 countries that have the largest number of MAC devices used are United States, China, Japan, Republic of Korea and Germany.

2.4 Propose and try an interesting statistical test or machine learning model you could use to gain insight from this dataset. Note, you don't have to use Machine Learning for this question. You can apply any analysis to the data even using SparkSQL, Python visualization libraries to analyze the data. Another example cloud be to apply correlation functions or other Spark functions to analyze the data. (2 marks)

NOTE: You may use MLLib in 2.4: https://spark.apache.org/docs/latest/ml-guide.html (https://spark.apache.org/docs/latest/ml-guide.html). Marks are awarded for the idea and implementation of the test/ML model.

One interesting statistical test/analysis would be to predict the CO2 level reading on the IoT devices based on other sensor readings such as the temperature, humidity and battery level and categorical variables such as country, lcd and device name to better understand the relationship between these variables and CO2 for future interpretations, applications and development of IoT devices. Therefore the CO2 level in this case is my target, dependent variable.

I will use machine learning and employ a linear regression model for this test. Steps:

1. Treat missing values.
2. Convert categorical variables to numerical.
3. Merge columns into single vector column.
4. Split the data into a training set and a test set.
5. Build and train linear regression model & evaluate the model on the test set.

1. Treat missing values (if any).

```
# Find count for Null, None, NaN of all columns in df, clean up data if necessary
from pyspark.sql.functions import col, isnan, when, count
df.select([count(when(isnan(c) | col(c).isNull(), c)).alias(c) for c in df.columns]).show()
```

```
+-------------+---------+----+----+---+---------+-----------+--------+---+--------+---+---------+-----+----+---------+
|battery_level|c02_level|cca2|cca3| cn|device_id|device_name|humidity| ip|latitude|lcd|longitude|scale|temp|timestamp|
+-------------+---------+----+----+---+---------+-----------+--------+---+--------+---+---------+-----+----+---------+
|            0|        0|   0|   0|  0|        0|          0|       0|  0|       0|  0|        0|    0|   0|        0|
+-------------+---------+----+----+---+---------+-----------+--------+---+--------+---+---------+-----+----+---------+
```

There are none (which is great for us!).

Some exploratory analysis...

```
df.describe().toPandas().transpose()
```

	0	1	2	3	4
summary	count	mean	stddev	min	max
battery_level	198164	4.4997678690377665	2.8733916884106137	0	9
c02_level	198164	1199.7639429967098	231.0600256290077	800	1599
cca2	198164	None	None	AD	ZW
cca3	198164	None	None	ABW	ZWE
cn	198164	None	None		Åland
device_id	198164	99082.5	57205.1637092317	1	198164
device_name	198164	None	None	device-mac-100005kiWNTBu	therm-stick-99995kaQv94y
humidity	198164	61.99212773258513	21.672313062313982	25	99
ip	198164	None	None	108.57.128.215	99.64.14.90
latitude	198164	36.521156062652814	17.90774071288935	-51.75	72.0
lcd	198164	None	None	green	yellow
longitude	198164	-0.6459595082860619	88.72758217920106	-175.0	178.42
scale	198164	None	None	Celsius	Celsius
temp	198164	22.012787388223895	7.209848253887109	10	34
timestamp	198164	1.4584440582462168E12	1708.115315742023	1458444054093	1458444061098

2. Convert categorical variables to numerical.

```
data_ml=df.drop("cca2","device_id","ip","latitude","longitude","scale","timestamp")
from pyspark.ml.feature import StringIndexer
indexer=StringIndexer(inputCol="cca3",outputCol="cca3_num")
indexed=indexer.fit(data_ml).transform(data_ml)
indexer=StringIndexer(inputCol="cn",outputCol="cn_num")
indexed=indexer.fit(indexed).transform(indexed)
indexer=StringIndexer(inputCol="device_name",outputCol="device_name_num")
indexed=indexer.fit(indexed).transform(indexed)
indexer=StringIndexer(inputCol="lcd",outputCol="lcd_num")
indexed=indexer.fit(indexed).transform(indexed)
```

```
indexed.show()
```

```
+-------------+---------+----+-------------+------------------+--------+------+----+--------+------+--------------+-------+
|battery_level|c02_level|cca3|           cn|       device_name|humidity|   lcd|temp|cca3_num|cn_num|device_name_num|lcd_num|
+-------------+---------+----+-------------+------------------+--------+------+----+--------+------+--------------+-------+
|            8|      868| USA|United States|meter-gauge-1xbYRYcj|      51| green|  34|     0.0|   0.0|       62173.0|    1.0|
```

```
|           7|     1473| NOR|       Norway|   sensor-pad-2n2Pea|      70|   red|  11|    19.0|  20.0|       146064.0|    2.0|
|           2|     1556| ITA|        Italy| device-mac-36TWSKiT|      44|   red|  19|    11.0|  11.0|        21359.0|    2.0|
|           6|     1080| USA|United States|   sensor-pad-4mzWkz|      32|yellow|  28|     0.0|   0.0|       157175.0|    0.0|
|           4|      931| PHL|  Philippines|therm-stick-5gimp...|      62| green|  25|    36.0|  37.0|       195199.0|    1.0|
|           3|     1210| USA|United States|sensor-pad-6al7RT...|      51|yellow|  27|     0.0|   0.0|       168286.0|    0.0|
|           3|     1129| CHN|        China|meter-gauge-7GeDoanM|      26|yellow|  18|     1.0|   1.0|        79950.0|    0.0|
|           0|     1536| JPN|        Japan|sensor-pad-8xUD6p...|      35|   red|  27|     2.0|   2.0|       179397.0|    2.0|
|           3|      807| JPN|        Japan| device-mac-9GcjZ2pw|      85| green|  13|     2.0|   2.0|        33026.0|    1.0|
|           7|     1470| USA|United States|sensor-pad-10Bsyw...|      56|   red|  26|     0.0|   0.0|        91426.0|    2.0|
|           3|     1544| ITA|        Italy|meter-gauge-11dlM...|      85|   red|  16|    11.0|  11.0|        38955.0|    2.0|
|           0|     1260| USA|United States|sensor-pad-12Y2kIm0o|      92|yellow|  12|     0.0|   0.0|       102537.0|    0.0|
|           6|     1007| IND|        India|meter-gauge-13Gro...|      92|yellow|  13|    17.0|  17.0|        44884.0|    0.0|
|           1|     1346| NOR|       Norway|sensor-pad-14QL93...|      90|yellow|  16|    19.0|  20.0|       113648.0|    0.0|
|           9|     1259| USA|United States|  device-mac-15se6mZ|      70|yellow|  13|     0.0|   0.0|        11110.0|    0.0|
|           4|     1425| USA|United States|sensor-pad-16aXmI...|      53|   red|  15|     0.0|   0.0|       124759.0|    2.0|
|           0|     1466| USA|United States|meter-gauge-17zb8...|      98|   red|  31|     0.0|   0.0|        56733.0|    2.0|
```

3. Merge columns into single vector column.

```
from pyspark.ml.linalg import Vector
from pyspark.ml.feature import VectorAssembler
assembler=VectorAssembler(inputCols=
['battery_level','cca3_num','cn_num','device_name_num','humidity','lcd_num','temp'],outputCol='features')
assembler
```

```
Out[28]: VectorAssembler_a19ecccda828
```

```
output=assembler.transform(indexed)
output.show()
```

```
+-------------+---------+----+-------------+--------------------+--------+------+----+--------+------+---------------+-------+-
------------------+
|battery_level|c02_level|cca3|           cn|         device_name|humidity|   lcd|temp|cca3_num|cn_num|device_name_num|lcd_num|
features|
+-------------+---------+----+-------------+--------------------+--------+------+----+--------+------+---------------+-------+-
------------------+
|            8|      868| USA|United States|meter-gauge-1xbYRYcj|      51| green|  34|     0.0|   0.0|        62173.0|    1.0|
[8.0,0.0,0.0,6217...|
|            7|     1473| NOR|       Norway|   sensor-pad-2n2Pea|      70|   red|  11|    19.0|  20.0|       146064.0|    2.0|
[7.0,19.0,20.0,14...|
|            2|     1556| ITA|        Italy| device-mac-36TWSKiT|      44|   red|  19|    11.0|  11.0|        21359.0|    2.0|
[2.0,11.0,11.0,21...|
|            6|     1080| USA|United States|   sensor-pad-4mzWkz|      32|yellow|  28|     0.0|   0.0|       157175.0|    0.0|
[6.0,0.0,0.0,1571...|
|            4|      931| PHL|  Philippines|therm-stick-5gimp...|      62| green|  25|    36.0|  37.0|       195199.0|    1.0|
[4.0,36.0,37.0,19...|
|            3|     1210| USA|United States|sensor-pad-6al7RT...|      51|yellow|  27|     0.0|   0.0|       168286.0|    0.0|
[3.0,0.0,0.0,1682...|
|            3|     1129| CHN|        China|meter-gauge-7GeDoanM|      26|yellow|  18|     1.0|   1.0|        79950.0|    0.0|
[3.0,1.0,1.0,7995...|
|            0|     1536| JPN|        Japan|sensor-pad-8xUD6p...|      35|   red|  27|     2.0|   2.0|       179397.0|    2.0|
```

```
output.select('features','c02_level').show()
```

```
+--------------------+---------+
|            features|c02_level|
+--------------------+---------+
|[8.0,0.0,0.0,6217...|      868|
|[7.0,19.0,20.0,14...|     1473|
|[2.0,11.0,11.0,21...|     1556|
|[6.0,0.0,0.0,1571...|     1080|
|[4.0,36.0,37.0,19...|      931|
|[3.0,0.0,0.0,1682...|     1210|
```

```
|[3.0,1.0,1.0,7995...|     1129|
|[0.0,2.0,2.0,1793...|     1536|
|[3.0,2.0,2.0,3302...|      807|
|[7.0,0.0,0.0,9142...|     1470|
|[3.0,11.0,11.0,38...|     1544|
|(7,[3,4,6],[10253...|     1260|
|[6.0,17.0,17.0,44...|     1007|
|[1.0,19.0,20.0,11...|     1346|
|[9.0,0.0,0.0,1111...|     1259|
|[4.0,0.0,0.0,1247...|     1425|
|[0.0,0.0,0.0,5673...|     1466|
```

4. Split the data into a training set and a test set.

```
final_data=output.select('features','c02_level')
```

```
train_data,test_data=final_data.randomSplit([0.75,0.25])
```

```
train_data.describe().show()
```

```
+-------+------------------+
|summary|         c02_level|
+-------+------------------+
|  count|            148422|
|   mean|1199.9690679279352|
| stddev|231.33613168992784|
|    min|               800|
|    max|              1599|
+-------+------------------+
```

```
test_data.describe().show()
```

```
+-------+------------------+
|summary|         c02_level|
+-------+------------------+
|  count|             49742|
|   mean| 1199.151883719995|
| stddev|230.23543592204206|
|    min|               800|
|    max|              1599|
+-------+------------------+
```

5. Build and train linear regression model & evaluate the model.

```
from pyspark.ml.regression import LinearRegression
lrobject=LinearRegression(featuresCol='features',labelCol='c02_level')
trained_model=lrobject.fit(train_data)
ship_results=trained_model.evaluate(train_data)
print('Rsquared Error:',ship_results.r2)
```

```
Rsquared Error: 0.15206225275773044
```

Rsquared Error is quite low i.e. 15% which means the data doesn't fit very well in the regression model. This also indicates the independent variables used are not explaining the variation of the dependent variable CO2 level much.

```
pred=trained_model.evaluate(test_data)
pred.predictions.show()
```

```
+--------------------+---------+------------------+
|            features|c02_level|        prediction|
+--------------------+---------+------------------+
|(7,[3,4,6],[2787....|     1224|1119.5925451821047|
|(7,[3,4,6],[7242....|     1187|1119.7512267558723|
|(7,[3,4,6],[10319...|     1050|1117.3117617336516|
|(7,[3,4,6],[12753...|     1070| 1117.619231484922|
|(7,[3,4,6],[14720...|     1030|1118.8647279620432|
|(7,[3,4,6],[15231...|     1393|1119.1025350337775|
|(7,[3,4,6],[17776...|     1329| 1118.624499506532|
|(7,[3,4,6],[18309...|     1248|1119.2217739434407|
|(7,[3,4,6],[18403...|     1224|1117.9403280152787|
|(7,[3,4,6],[18488...|     1123|1117.1450927458418|
|(7,[3,4,6],[18489...|     1145|1116.2375912039772|
|(7,[3,4,6],[18558...|     1004|1117.6510894288465|
|(7,[3,4,6],[19051...|     1078|1117.7342930816226|
|(7,[3,4,6],[22869...|     1198|1116.3019548746613|
|(7,[3,4,6],[22968...|     1278|   1118.4636439301|
|(7,[3,4,6],[23214...|     1172| 1118.118713887576|
|(7,[3,4,6],[23958...|     1138|1118.9945699390225|
|(7,[3,4,6],[24303...|     1045|1118.4122908754614|
```

The predicted values vary a lot from actual CO2 levels.

I then tried Random Forest as it is a model better suited for non-linear data but ran into an error at the step of training the model.

```
#from pyspark.ml.classification import RandomForestClassifier

#rf = RandomForestClassifier(featuresCol='features',labelCol='c02_level')
#rfModel = rf.fit(train_data)
#pred2 = rfModel.transform(test_data)
#pred2.predictions.show()
```

Next I tried a subset of the data using linear regression again, to see if it could yield better results, i.e. United States only. I am thinking maybe there are identifiable patterns within geographical regions.

```
output.show()
```

```
+-------------+---------+----+-------------+--------------------+--------+------+----+--------+------+--------------+-------+-
------------------+
|battery_level|c02_level|cca3|           cn|         device_name|humidity|   lcd|temp|cca3_num|cn_num|device_name_num|lcd_num|
features|
+-------------+---------+----+-------------+--------------------+--------+------+----+--------+------+--------------+-------+-
------------------+
|            8|      868| USA|United States|meter-gauge-1xbYRYcj|      51| green|  34|     0.0|   0.0|        62173.0|    1.0|
[8.0,0.0,0.0,6217...|
|            7|     1473| NOR|       Norway|   sensor-pad-2n2Pea|      70|   red|  11|    19.0|  20.0|       146064.0|    2.0|
[7.0,19.0,20.0,14...|
|            2|     1556| ITA|        Italy| device-mac-36TWSKiT|      44|   red|  19|    11.0|  11.0|        21359.0|    2.0|
[2.0,11.0,11.0,21...|
|            6|     1080| USA|United States|   sensor-pad-4mzWkz|      32|yellow|  28|     0.0|   0.0|       157175.0|    0.0|
[6.0,0.0,0.0,1571...|
```

```
|           4|      931| PHL|  Philippines|therm-stick-5gimp...|      62| green|  25|    36.0|  37.0|       195199.0|    1.0|
[4.0,36.0,37.0,19...|
|           3|     1210| USA|United States|sensor-pad-6al7RT...|      51|yellow|  27|     0.0|   0.0|       168286.0|    0.0|
[3.0,0.0,0.0,1682...|
|           3|     1129| CHN|        China|meter-gauge-7GeDoanM|      26|yellow|  18|     1.0|   1.0|        79950.0|    0.0|
[3.0,1.0,1.0,7995...|
|           0|     1536| JPN|        Japan|sensor-pad-8xUD6p...|      35|   red|  27|     2.0|   2.0|       179397.0|    2.0|
```

```
df_us=output.filter(df.cn=='United States')
df_us.show()
```

```
+-------------+---------+----+-------------+--------------------+--------+------+----+--------+------+--------------+-------+-
------------------+
|battery_level|c02_level|cca3|           cn|         device_name|humidity|   lcd|temp|cca3_num|cn_num|device_name_num|lcd_num|
features|
+-------------+---------+----+-------------+--------------------+--------+------+----+--------+------+--------------+-------+-
------------------+
|           8|      868| USA|United States|meter-gauge-1xbYRYcj|      51| green|  34|     0.0|   0.0|        62173.0|    1.0|
[8.0,0.0,0.0,6217...|
|           6|     1080| USA|United States|   sensor-pad-4mzWkz|      32|yellow|  28|     0.0|   0.0|       157175.0|    0.0|
[6.0,0.0,0.0,1571...|
|           3|     1210| USA|United States|sensor-pad-6al7RT...|      51|yellow|  27|     0.0|   0.0|       168286.0|    0.0|
[3.0,0.0,0.0,1682...|
|           7|     1470| USA|United States|sensor-pad-10Bsyw...|      56|   red|  26|     0.0|   0.0|        91426.0|    2.0|
[7.0,0.0,0.0,9142...|
|           0|     1260| USA|United States|sensor-pad-12Y2kIm0o|      92|yellow|  12|     0.0|   0.0|       102537.0|    0.0|
(7,[3,4,6],[10253...|
|           9|     1259| USA|United States|  device-mac-15se6mZ|      70|yellow|  13|     0.0|   0.0|        11110.0|    0.0|
[9.0,0.0,0.0,1111...|
|           4|     1425| USA|United States|sensor-pad-16aXmI...|      53|   red|  15|     0.0|   0.0|       124759.0|    2.0|
[4.0,0.0,0.0,1247...|
|           0|     1466| USA|United States|meter-gauge-17zb8...|      98|   red|  31|     0.0|   0.0|        56733.0|    2.0|
```

```
df_us.select('features','c02_level').show()
```

```
+--------------------+---------+
|            features|c02_level|
+--------------------+---------+
|[8.0,0.0,0.0,6217...|      868|
|[6.0,0.0,0.0,1571...|     1080|
|[3.0,0.0,0.0,1682...|     1210|
|[7.0,0.0,0.0,9142...|     1470|
|(7,[3,4,6],[10253...|     1260|
|[9.0,0.0,0.0,1111...|     1259|
|[4.0,0.0,0.0,1247...|     1425|
|[0.0,0.0,0.0,5673...|     1466|
|[9.0,0.0,0.0,6217...|     1531|
|[7.0,0.0,0.0,1410...|     1155|
|[4.0,0.0,0.0,1926...|      880|
|[6.0,0.0,0.0,1466...|      992|
|[8.0,0.0,0.0,1477...|     1262|
|[8.0,0.0,0.0,1555...|      857|
|[4.0,0.0,0.0,7106...|      967|
|[2.0,0.0,0.0,1588...|     1032|
|[8.0,0.0,0.0,1594...|     1436|
|[6.0,0.0,0.0,2524...|     1503|
```

```
final_data=df_us.select('features','c02_level')
```

```
train_data,test_data=final_data.randomSplit([0.75,0.25])
```

```
train_data.describe().show()
```

```
+-------+------------------+
|summary|         c02_level|
+-------+------------------+
|  count|             51560|
|   mean| 1200.878975950349|
| stddev|230.92615414358028|
|    min|               800|
|    max|              1599|
+-------+------------------+
```

```
test_data.describe().show()
```

```
+-------+------------------+
|summary|         c02_level|
+-------+------------------+
|  count|             16985|
|   mean|1198.3170444509863|
| stddev| 230.3485114934973|
|    min|               800|
|    max|              1599|
+-------+------------------+
```

```
from pyspark.ml.regression import LinearRegression
lrobject=LinearRegression(featuresCol='features',labelCol='c02_level')
trained_model=lrobject.fit(train_data)
ship_results=trained_model.evaluate(train_data)
print('Rsquared Error:',ship_results.r2)
```

```
Rsquared Error: 0.15480162101757544
```

Rsquared value is still 15% with a smaller set.

```
+--------------------+---------+------------------+
|            features|c02_level|        prediction|
+--------------------+---------+------------------+
|(7,[3,4,6],[2787....|     1224|1119.7425032445585|
|(7,[3,4,6],[4430....|     1321|1115.5445861684168|
|(7,[3,4,6],[8686....|     1212|1118.6177122723752|
|(7,[3,4,6],[10985...|     1217|1115.9688299294412|
|(7,[3,4,6],[12376...|     1125|1115.5696054856796|
|(7,[3,4,6],[17539...|     1232|1116.9345846229437|
|(7,[3,4,6],[17776...|     1329|1118.8496944786107|
|(7,[3,4,6],[18309...|     1248|1119.3833356702871|
|(7,[3,4,6],[18357...|     1114|1118.4025385524847|
|(7,[3,4,6],[18538...|     1083|1115.9608450355643|
|(7,[3,4,6],[18693...|     1338| 1115.916687242617|
|(7,[3,4,6],[18907...|     1195|1119.7601974105278|
|(7,[3,4,6],[19144...|     1122|1115.5802610626351|
|(7,[3,4,6],[19360...|     1001| 1116.504607197079|
|(7,[3,4,6],[19387...|     1213|1117.9373216506342|
|(7,[3,4,6],[19913...|     1062|1119.7681729089518|
|(7,[3,4,6],[21692...|     1057|1118.4167534655928|
|(7,[3,4,6],[24778...|     1152|1119.4216331969724|
```