

BigData Analytics to support the decision making for solving the flooding problem
flooding_project@2 FINISHED

การประมวลผลข้อมูล Bigdata เพื่อสนับสนุนการตัดสินใจวางแผนแก้ปัญหาน้ำท่วมในพื้นที่กรุงเทพฯ

รายชื่อสมาชิกในกลุ่ม

- 62130700311 ຄນພັດນີ້ ຈັດຕະວາງ
 - 62130700314 ລັກຄນາ ຕົກແສງຮຽມ
 - 62130700315 ວິຊາຮຽນ ຈາກແສງ
 - 62130700329 ຄນວັດນີ້ ສຸການທີ່

Took 3 sec. Last updated by anonymous at November 29 2020, 9:26:56 PM. (outdated)

บทนำ

FINISHED

ประเทศไทยยังคงมีความต้องการที่จะซื้อสินค้าและบริการจากต่างประเทศอย่างต่อเนื่อง

กรุงเทพมหานครนั้น ตั้งอยู่ในพื้นที่ราบลุ่มตอนปลายของแม่น้ำเจ้าพระยา ซึ่งอยู่กลางไดอีทธิพลการซึ่น-ลงของระดับน้ำทะเลในอ่าวไทย ด้วยวิธีชี้วัดในอดีต ของประชากรในพื้นที่กรุงเทพมหานคร นิยมเดินทางโดยใช้เรือ ส่งผลให้ลักษณะภูมิศาสตร์ของกรุงเทพมหานครมีแม่น้ำ ลำคลองจำนวนมากกระจายอยู่ ตามพื้นที่ต่างๆ ซึ่งปัจจุบันเหลือเพียงหนึ่งเท่านั้นที่เป็นจุดระบายน้ำปรimitation น้ำลงไปสู่ท่าเรือทางอ่าวไทย

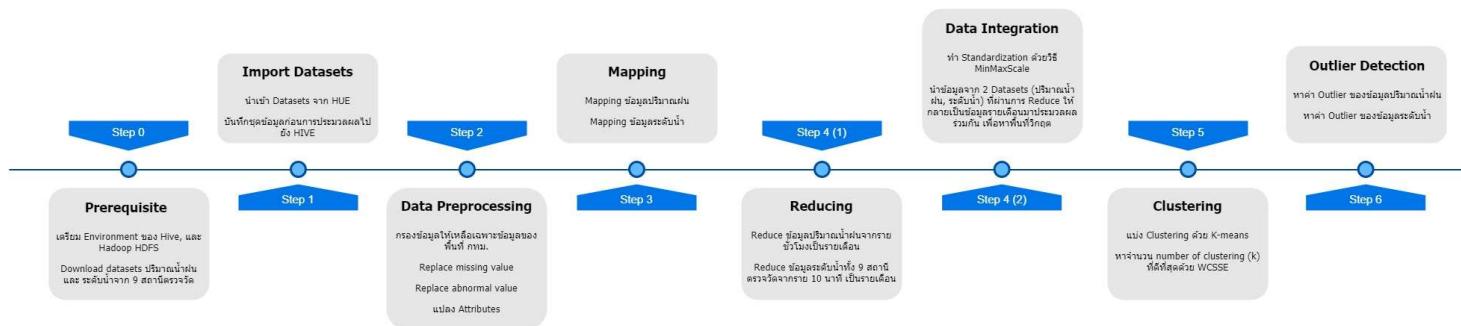
สิ่งที่น่าสนใจคือ ทำไม่เมืองที่มีแม่น้ำลากคลองน้อยใหญ่มากมายกระจาดเดือดอยู่ตามพื้นที่ต่างๆ และกลับประสบปัญหาน้ำท่วมซึ่งเมื่อผ่านตก สิ่งนี้แสดงให้เห็นว่าเชิงเพมาน้านรักการกำลังประสมปัญหาน้ำเกี่ยวกับการระบายน้ำจากถนนไปยังแหล่งน้ำที่ยังไม่ทำได้ในเดียว ซึ่งหน่วยงานภาครัฐมีความพยายามในการแก้ไขปัญหาน้ำ ซึ่งวิธีแก้ไขปัญหาน้ำที่พบได้บ่อยมากที่สุดคือการทารุณ化ท่อระบายน้ำ หรือการ “ลอกท่อ”

ทางกลุ่มผู้จัดทำให้นิวน่าหากใช้ ข้อมูลปริมาณน้ำฝน และ ข้อมูลระดับน้ำจากทั้ง 9 สถานีตรวจวัดรอบพื้นที่กรุงเทพมหานคร มาประมวลผลและวิเคราะห์รวมกัน จะสามารถช่วยสนับสนุนการตัดสินใจได้ว่า ในแต่ละเดือนพื้นที่ไหนจะมีโอกาสอยู่ในภาวะวิกฤตควรรับดำเนินการ “ลอกห้อ” และพื้นที่ไหนที่ยังสามารถรับน้ำได้ดีอยู่ไม่ต้องการ “ลอกห้อ” เพื่อเป็นการวางแผนในช่วงหน้ากัดประลิทธิภาร

Took 0 sec. Last updated by anonymous at November 29 2020, 9:26:56 PM. (outdated)

ขั้นตอนการประเมินผล

FINISHED



กรุณา Ctrl+Click ที่ link เพื่อดูรูปด้านลับ link (<https://raw.githubusercontent.com/TanaphatC/bigdata-flooding-project/master/images/processing.jpg>)

Took 0 sec. | Last updated by anonymous at November 29 2020, 9:26:56 PM (outdated)

Step 0: Prequisite

FINISHED

- 0.1) สร้าง Database 'flooding_db' ที่ /Hive
 - 0.2) สร้าง workspace folder '/user/zeppelin/flooding_data' ที่ HDFS

0.1) Prerequisite - Prepare Hive Database at location '/hive'

```
%sql  
CREATE DATABASE IF NOT EXISTS flooding_db COMMENT 'This is  
flooding project database' LOCATION '/hive'
```

Took 0 sec. Last updated by anonymous at November 29 2020, 9:44:59 PM. (outdated)

FINISHED

0.2) Prerequisite - Prepare Hadoop HDFS environment

```
%sh  
if hdfs dfs -test -e "/user/zeppelin/flooding_data"; then  
    hdfs dfs -rm -r /user/zeppelin/flooding_data  
else  
    echo "/user/zeppelin/flooding_data is existing"  
fi  
  
/user/zeppelin/flooding_data is existing
```

Took 5 sec. Last updated by anonymous at November 29 2020, 9:27:31 PM. (outdated)

FINISHED

0.3) Prerequisite - Prepare Hive environment

```
%pyspark  
from pyspark.sql import HiveContext  
  
sqlContext = HiveContext(sc)  
tableCatalog = spark._jsparkSession.catalog()  
if tableCatalog.tableExists('flooding_db', 'rainyHourly'):  
    sqlContext.sql("drop table flooding_db.rainyHourly")  
    print "Drop table rainyHourly successful"  
  
if tableCatalog.tableExists('flooding_db', 'waterLvBKK001'):  
    sqlContext.sql("drop table flooding_db.waterLvBKK001")  
    print "Drop table waterLvBKK001 successful"  
  
if tableCatalog.tableExists('flooding_db', 'waterLvBKK003'):  
    sqlContext.sql("drop table flooding_db.waterLvBKK003")  
    print "Drop table waterLvBKK003 successful"  
  
if tableCatalog.tableExists('flooding_db', 'waterLvBKK004'):  
    sqlContext.sql("drop table flooding_db.waterLvBKK004")  
    print "Drop table waterLvBKK004 successful"  
  
if tableCatalog.tableExists('flooding_db', 'waterLvBKK005'):  
    sqlContext.sql("drop table flooding_db.waterLvBKK005")  
    print "Drop table waterLvBKK005 successful"  
  
if tableCatalog.tableExists('flooding_db', 'waterLvBKK008'):  
    sqlContext.sql("drop table flooding_db.waterLvBKK008")  
    print "Drop table waterLvBKK008 successful"  
  
if tableCatalog.tableExists('flooding_db', 'waterLvBKK009'):  
    sqlContext.sql("drop table flooding_db.waterLvBKK009")  
    print "Drop table waterLvBKK009 successful"  
  
if tableCatalog.tableExists('flooding_db',  
    'waterLvBKK0020'):  
    sqlContext.sql("drop table flooding_db.waterLvBKK0020")  
    print "Drop table waterLvBKK0020 successful"  
  
if tableCatalog.tableExists('flooding_db',  
    'waterLvBKK0021'):  
    sqlContext.sql("drop table flooding_db.waterLvBKK0021")  
    print "Drop table waterLvBKK0021 successful"  
  
if tableCatalog.tableExists('flooding_db',  
    'waterLvCPY0015'):  
    sqlContext.sql("drop table flooding_db.waterLvCPY0015")  
    print "Drop table waterLvCPY0015 successful"  
  
if tableCatalog.tableExists('flooding_db', 'criticalScore'):  
    sqlContext.sql("drop table flooding_db.criticalScore")  
    print "Drop table criticalScore successful"  
  
Drop table rainyHourly successful  
Drop table waterLvBKK001 successful  
Drop table waterLvBKK003 successful  
Drop table waterLvBKK004 successful  
Drop table waterLvBKK005 successful  
Drop table waterLvBKK008 successful  
Drop table waterLvBKK009 successful  
Drop table waterLvBKK0020 successful  
Drop table waterLvBKK0021 successful  
Drop table waterLvCPY0015 successful
```

```
DataFrame[]  
Drop table criticalScore successful
```

Took 8 sec. Last updated by anonymous at November 29 2020, 9:27:39 PM. (outdated)

flooding_project

FINISHED

Prequisite

0.4) Download ข้อมูลปริมาณน้ำฝนจาก Github และ upload ลงไปที่ folder '/user/zeppelin/flooding_data' ใน Hadoop HDFS

ข้อมูลปริมาณน้ำฝนได้มากจากข้อมูลน้ำฝนรวมรายชั่วโมง(มีลิสเมตอร์) จากระบบ Telemetering กรมอุตุนิยมวิทยา(DGA Open Government, 2558)

ข้อมูลมี Meta Data ดังนี้

รายการข้อมูล	คำอธิบาย
Station_id	รหัสสถานี
Station_Name	ชื่อสถานี
Location	ที่ตั้ง
Province_Name	จังหวัด
Latitude	ละติจูด(N)
Longitude	ลองติจูด(E)
Date	วันที่
H00_01	ปริมาณฝนเวลา 00:01น.
H01_02	ปริมาณฝนเวลา 01:02น.
H02_03	ปริมาณฝนเวลา 02:03น.
H03_04	ปริมาณฝนเวลา 03:04น.
H04_05	ปริมาณฝนเวลา 04:05น.
H05_06	ปริมาณฝนเวลา 05:06น.
H06_07	ปริมาณฝนเวลา 06:07น.
H07_08	ปริมาณฝนเวลา 07:08น.
H08_09	ปริมาณฝนเวลา 08:09น.
H09_10	ปริมาณฝนเวลา 09:10น.
H10_11	ปริมาณฝนเวลา 10:11น.
H11_12	ปริมาณฝนเวลา 11:12น.
H12_13	ปริมาณฝนเวลา 12:13น.
H13_14	ปริมาณฝนเวลา 13:14น.
H14_15	ปริมาณฝนเวลา 14:15น.
H15_16	ปริมาณฝนเวลา 15:16น.
H16_17	ปริมาณฝนเวลา 16:17น.
H17_18	ปริมาณฝนเวลา 17:18น.
H18_19	ปริมาณฝนเวลา 18:19น.
H19_20	ปริมาณฝนเวลา 19:20น.
H20_21	ปริมาณฝนเวลา 20:21น.
H21_22	ปริมาณฝนเวลา 21:22น.

รายการข้อมูล	คำอธิบาย
H22_23	ปริมาณฝนเวลา 22:23น.
H23_24	ปริมาณฝนเวลา 23:24น.

Took 0 sec. Last updated by anonymous at November 29 2020, 9:27:39 PM. (outdated)

Prerequisite

FINISHED

0.5) และ 0.6) - Download ข้อมูลระดับน้ำจากทั้ง 9 สถานีตรวจดรอปพื้นที่กรุงเทพมหานคร จาก Github และ upload ลงไปที่ folder '/user/zeppelin/flooding_data' ใน Hadoop HDFS

ข้อมูลระดับน้ำได้มากจากชุดข้อมูลระดับน้ำจากระบบโทรมาตรหัวประเทศไทยของสถาบันสารสนเทศทรัพยากรน้ำ (DGA Open Government, 2563)

ข้อมูลเมื่อ Meta Data ดังนี้

รายการข้อมูล	คำอธิบาย
code	รหัสสถานี
geocode	รหัสไปรษณีย์
name	ชื่อสถานีภาษาไทย
lat	ละติจูด
long	ลองจิจูด
prov_id	รหัสจังหวัด
tambon_name	ชื่อตำบลภาษาไทย
amphoe_name	ชื่ออำเภอภาษาไทย
province_name	ชื่อจังหวัดภาษาไทย
basin	ชื่อสูบน้ำภาษาไทย
name_e	ชื่อสถานีภาษาอังกฤษ
tambon_name_e	ชื่อตำบลภาษาอังกฤษ
amphoe_name_e	ชื่ออำเภอภาษาอังกฤษ
province_name_e	ชื่อจังหวัดภาษาอังกฤษ
wl_offset	ค่า offset ของระดับน้ำ
station_type	ประเภทของสถานี
left_bank	ระดับตลิ่งซ้าย
right_bank	ระดับตลิ่งขวา

Took 0 sec. Last updated by anonymous at November 29 2020, 9:27:39 PM. (outdated)

อ้างอิงข้อมูลจาก:

FINISHED

DGA Open Government. (2558). ข้อมูลน้ำฝนรวมรายชั่วโมง ปี 2014 [ออนไลน์]. แหล่งที่มา:
https://opendata.data.go.th/en/dataset/item_e9ab5b50-d228-4953-ac6f-aeeed71ec751
[\[https://opendata.data.go.th/en/dataset/item_e9ab5b50-d228-4953-ac6f-aeeed71ec751\] \[25 ตุลาคม 2563\]](https://opendata.data.go.th/en/dataset/item_e9ab5b50-d228-4953-ac6f-aeeed71ec751)

DGA Open Government. (2563). ข้อมูลระดับน้ำราย 10 นาที ปี 2014-2018 [ออนไลน์]. แหล่งที่มา:
<https://www.data.go.th/en/dataset/set-of-water-level-by-station> (<https://www.data.go.th/en/dataset/set-of-water-level-by-station>)
[\[17 ตุลาคม 2563\]](#)

Took 0 sec. Last updated by anonymous at November 29 2020, 9:27:40 PM. (outdated)

0.4) Prerequisite - Download rainy rate datasets from Github to Hadoop HDFS

FINISHED

```
%sh
if ! hdfs dfs -test -e "/user/zeppelin/flooding_data"; then
    hdfs dfs -mkdir -p /user/zeppelin/flooding_data
fi
echo "prepare working directory /user/zeppelin/flooding_data"
if ! hdfs dfs -test -e "/user/zeppelin/flooding_data/rainfall_hourly_data2.csv"; then
    wget https://github.com/TanaphatC/bigdata-flooding-project/blob/master/rainfall_hourly_data2.csv?raw=true -O rainfall_hourly_data2.csv
    hdfs dfs -put rainfall_hourly_data2.csv /user/zeppelin/flooding_data
fi
echo "Download Hourly Rainy Rate dataset complete"

26550K ..... 97% 159M 0s
26600K ..... 97% 164M 0s
26650K ..... 97% 147M 0s
26700K ..... 97% 168M 0s
26750K ..... 97% 167M 0s
26800K ..... 98% 170M 0s
26850K ..... 98% 144M 0s
26900K ..... 98% 135M 0s
26950K ..... 98% 164M 0s
27000K ..... 98% 140M 0s
27050K ..... 99% 133M 0s
27100K ..... 99% 112M 0s
27150K ..... 99% 154M 0s
27200K ..... 99% 70.2M 0s
27250K ..... 99% 60.3M 0s
27300K ..... 99% 171M 0s
27350K ..... 100% 141M=0.6s
```

Took 10 sec. Last updated by anonymous at November 29 2020, 9:27:50 PM. (outdated)

0.5) Prerequisite - Download datasets from Github to Hadoop HDFS part 1

FINISHED

```
%sh
if ! hdfs dfs -test -e "/user/zeppelin/flooding_data/water_level_station_BKK001.csv"; then
    wget https://github.com/TanaphatC/bigdata-flooding-project/blob/master/water_level_station_BKK001.csv?raw=true -O water_level_station_BKK001.csv
    hdfs dfs -put water_level_station_BKK001.csv /user/zeppelin/flooding_data
fi
echo "Download water-level dataset of station 'BKK001' complete"

if ! hdfs dfs -test -e "/user/zeppelin/flooding_data/water_level_station_BKK003.csv"; then
    wget https://github.com/TanaphatC/bigdata-flooding-project/blob/master/water_level_station_BKK003.csv?raw=true -O water_level_station_BKK003.csv
    hdfs dfs -put water_level_station_BKK003.csv /user/zeppelin/flooding_data
fi
echo "Download water-level dataset of station 'BKK003' complete"

if ! hdfs dfs -test -e "/user/zeppelin/flooding_data/water_level_station_BKK004.csv"; then
    wget https://github.com/TanaphatC/bigdata-flooding-project/blob/master/water_level_station_BKK004.csv?raw=true -O water_level_station_BKK004.csv
    hdfs dfs -put water_level_station_BKK004.csv /user/zeppelin/flooding_data
fi
echo "Download water-level dataset of station 'BKK004' complete"

if ! hdfs dfs -test -e "/user/zeppelin/flooding_data/water_level_station_BKK005.csv"; then
    wget https://github.com/TanaphatC/bigdata-flooding-project/blob/master/water_level_station_BKK005.csv?raw=true -O water_level_station_BKK005.csv
    hdfs dfs -put water_level_station_BKK005.csv /user/zeppelin/flooding_data
fi
echo "Download water-level dataset of station 'BKK005' complete"

if ! hdfs dfs -test -e "/user/zeppelin/flooding_data/water_level_station_BKK008.csv"; then
    wget https://github.com/TanaphatC/bigdata-flooding-project/blob/master/water_level_station_BKK008.csv?raw=true -O water_level_station_BKK008.csv
    hdfs dfs -put water_level_station_BKK008.csv /user/zeppelin/flooding_data
fi
echo "Download water-level dataset of station 'BKK008' complete"

5850K ..... 90% 20.1M 0s
5900K ..... 91% 85.2M 0s
5950K ..... 91% 158M 0s
6000K ..... 92% 44.9M 0s
6050K ..... 93% 51.5M 0s
6100K ..... 94% 35.0M 0s
6150K ..... 94% 83.2M 0s
6200K ..... 95% 58.9M 0s
6250K ..... 96% 37.5M 0s
6300K ..... 97% 67.5M 0s
6350K ..... 97% 21.0M 0s
6400K ..... 98% 92.8M 0s
6450K ..... 99% 111M 0s
6500K ..... 100% 45.6M=0.2s
```

2020-11-29 14:28:16 (30.8 MB/s) - 'water_level_station_BKK008.csv' saved [6694110/6694110]

Download water-level dataset of station 'BKK008' complete

Took 28 sec. Last updated by anonymous at November 29 2020, 9:28:18 PM. (outdated)

0.6 Prerequisite - Download datasets from Github to Hadoop HDFS part 2

FINISHED

```
%sh
if hdfs dfs -test -e "/user/zeppelin/flooding_data/water_level_station_BKK009.csv"; then
    wget https://github.com/TanaphatC/bigdata-flooding-project/blob/master
        /water_level_station_BKK009.csv?raw=true -O water_level_station_BKK009.csv
    hdfs dfs -put water_level_station_BKK009.csv /user/zeppelin/flooding_data
fi
echo "Download water-level dataset of station 'BKK009' complete"

if ! hdfs dfs -test -e "/user/zeppelin/flooding_data/water_level_station_BKK020.csv"; then
    wget https://github.com/TanaphatC/bigdata-flooding-project/blob/master
        /water_level_station_BKK020.csv?raw=true -O water_level_station_BKK020.csv
    hdfs dfs -put water_level_station_BKK020.csv /user/zeppelin/flooding_data
fi
echo "Download water-level dataset of station 'BKK020' complete"

if ! hdfs dfs -test -e "/user/zeppelin/flooding_data/water_level_station_BKK021.csv"; then
    wget https://github.com/TanaphatC/bigdata-flooding-project/blob/master
        /water_level_station_BKK021.csv?raw=true -O water_level_station_BKK021.csv
    hdfs dfs -put water_level_station_BKK021.csv /user/zeppelin/flooding_data
fi
echo "Download water-level dataset of station 'BKK021' complete"

if ! hdfs dfs -test -e "/user/zeppelin/flooding_data/water_level_station_CPY015.csv"; then
    wget https://github.com/TanaphatC/bigdata-flooding-project/blob/master
        /water_level_station_CPY015.csv?raw=true -O water_level_station_CPY015.csv
    hdfs dfs -put water_level_station_CPY015.csv /user/zeppelin/flooding_data
fi
echo "Download water-level dataset of station 'CPY015' complete"

5750K ..... 89% 50.7M 0s
5800K ..... 90% 154M 0s
5850K ..... 91% 46.5M 0s
5900K ..... 92% 45.1M 0s
5950K ..... 93% 45.9M 0s
6000K ..... 93% 45.7M 0s
6050K ..... 94% 151M 0s
6100K ..... 95% 53.2M 0s
6150K ..... 96% 18.2M 0s
6200K ..... 96% 95.0M 0s
6250K ..... 97% 41.6M 0s
6300K ..... 98% 179M 0s
6350K ..... 99% 50.2M 0s
6400K ..... 100% 38.1M=0.2s
```

2020-11-29 14:28:39 (29.6 MB/s) - ‘water_level_station_CPY015.csv’ saved [6599582/6599582]

Download water-level dataset of station 'CPY015' complete

Took 23 sec. Last updated by anonymous at November 29 2020, 9:28:41 PM. (outdated)

Step 1: Import Data

FINISHED

Took 0 sec. Last updated by anonymous at November 29 2020, 9:28:41 PM. (outdated)

1.1) นำเข้าข้อมูลปริมาณน้ำฝนจาก HUE สร้างเป็น Dataframe

FINISHED

1.2) นำเข้าข้อมูลระดับน้ำจาก 9 สถานีวัดจาก HUE และสร้างเป็น Dataframe ของแต่ละสถานีวัด โดยข้อมูลสถานีที่นำมาใช้ในการวิเคราะห์ครั้งนี้ได้แก่

รหัสสถานี	ชื่อสถานี	อำเภอ	ตำบล
BKK001	คลองลาดพร้าว ท้ายบptr.คลอง2	สายไหม	สายไหม
BKK003	คลองมหาสวัสดิ์ บางกรวย-สวนผัก	ตลึงชัน	ตลึงชัน
BKK004	คลองทวีวัฒนา ท้ายบptr.ทวีวัฒนา	ศalaธรรมสพน์	ทวีวัฒนา
BKK005	คลองภาษีเจริญ เพชรเกษม69	หลักสอง	บางแค
BKK008	คลองแ سنแสน บางกะปี	หัวหมาก	บางกะปี
BKK009	คลองล้ำปลาทิว ลาดกระบัง	ลาดกระบัง	ลาดกระบัง
BKK020	คลองลาดพร้าว ปากคลอง2สายใต้	ลาดพร้าว	ลาดพร้าว
BKK021	คลองลาดพร้าว วัดบางบัว	อนุสาวรีย์	บางเขน
CPY015	สะพานกรุงเทพ	ดาวคะนอง	ธนบุรี

1.3) បង្កើតអំពីអាមេរិកនៃការបរមាបលទិវ HIVE

Took 0 sec. Last updated by anonymous at November 29 2020, 9:28:41 PM. (outdated)

flooding project

1.1) Hourly rainfall rate dataset importing from HUE

SPARK JOBS FINISHED

```
%pyspark
hourlyData = spark.read.csv("/user/zeppelin/flooding_data/rainfall_hourly_data2.csv", header=True,
                             inferSchema=True)
print "Import 'Hourly Rainy Rate' dataset successful"

Import 'Hourly Rainy Rate' dataset successful
```

Took 8 sec. Last updated by anonymous at November 29 2020, 9:28:50 PM. (outdated)

1.2) Hourly water level rate dataset importing from HUE

SPARK JOBS FINISHED

```
%pyspark
from pyspark.sql.functions import lit

waterLevelBKK001DF = spark.read.csv("/user/zeppelin/flooding_data/water_level_station_BKK001.csv", header
                                     =True, inferSchema=True).withColumn("station", lit("BKK001"))
print "Import 'Water Level of station BKK001' dataset successful"

waterLevelBKK003DF = spark.read.csv("/user/zeppelin/flooding_data/water_level_station_BKK003.csv", header
                                     =True, inferSchema=True).withColumn("station", lit("BKK003"))
print "Import 'Water Level of station BKK003' dataset successful"

waterLevelBKK004DF = spark.read.csv("/user/zeppelin/flooding_data/water_level_station_BKK004.csv", header
                                     =True, inferSchema=True).withColumn("station", lit("BKK004"))
print "Import 'Water Level of station BKK004' dataset successful"

waterLevelBKK005DF = spark.read.csv("/user/zeppelin/flooding_data/water_level_station_BKK005.csv", header
                                     =True, inferSchema=True).withColumn("station", lit("BKK005"))
print "Import 'Water Level of station BKK005' dataset successful"

waterLevelBKK008DF = spark.read.csv("/user/zeppelin/flooding_data/water_level_station_BKK008.csv", header
                                     =True, inferSchema=True).withColumn("station", lit("BKK008"))
print "Import 'Water Level of station BKK008' dataset successful"

waterLevelBKK009DF = spark.read.csv("/user/zeppelin/flooding_data/water_level_station_BKK009.csv", header
                                     =True, inferSchema=True).withColumn("station", lit("BKK009"))
print "Import 'Water Level of station BKK009' dataset successful"

waterLevelBKK020DF = spark.read.csv("/user/zeppelin/flooding_data/water_level_station_BKK020.csv", header
                                     =True, inferSchema=True).withColumn("station", lit("BKK020"))
print "Import 'Water Level of station BKK020' dataset successful"

waterLevelBKK021DF = spark.read.csv("/user/zeppelin/flooding_data/water_level_station_BKK021.csv", header
                                     =True, inferSchema=True).withColumn("station", lit("BKK021"))
print "Import 'Water Level of station BKK021' dataset successful"

waterLevelCPY015DF = spark.read.csv("/user/zeppelin/flooding_data/water_level_station_CPY015.csv", header
                                     =True, inferSchema=True).withColumn("station", lit("CPY015"))
print "Import 'Water Level of station CPY015' dataset successful"

Import 'Water Level of station BKK001' dataset successful
Import 'Water Level of station BKK003' dataset successful
Import 'Water Level of station BKK004' dataset successful
Import 'Water Level of station BKK005' dataset successful
Import 'Water Level of station BKK008' dataset successful
Import 'Water Level of station BKK009' dataset successful
Import 'Water Level of station BKK020' dataset successful
Import 'Water Level of station BKK021' dataset successful
Import 'Water Level of station CPY015' dataset successful
```

Took 20 sec. Last updated by anonymous at November 29 2020, 9:29:10 PM. (outdated)

1.3) Save preprocessing datasets to HIVE

SPARK JOBS FINISHED

```
%pyspark
tableCatalog = spark._jsparkSession.catalog()
if not tableCatalog.tableExists('flooding_db', 'rainyHourly'):
    hourlyData.write.saveAsTable("flooding_db.rainyHourly")

print "Save Hourly Rainy Rate before processing to HIVE successful"

if not tableCatalog.tableExists('flooding_db', 'waterLvBKK001'):
    waterLevelBKK001DF.write.saveAsTable("flooding_db.waterLvBKK001")

if not tableCatalog.tableExists('flooding_db', 'waterLvBKK003'):
    waterLevelBKK003DF.write.saveAsTable("flooding_db.waterLvBKK003")

if not tableCatalog.tableExists('flooding_db', 'waterLvBKK004'):
    waterLevelBKK004DF.write.saveAsTable("flooding_db.waterLvBKK004")

if not tableCatalog.tableExists('flooding_db', 'waterLvBKK005'):
    waterLevelBKK005DF.write.saveAsTable("flooding_db.waterLvBKK005")

if not tableCatalog.tableExists('flooding_db', 'waterLvBKK008'):
    waterLevelBKK008DF.write.saveAsTable("flooding_db.waterLvBKK008")

if not tableCatalog.tableExists('flooding_db', 'waterLvBKK009'):
```

```

waterLevelBKK009DF.write.saveAsTable("flooding_db.waterLvBKK009")

if not tableCatalog.tableExists('flooding_db', 'waterLvBKK0020'):
    waterLevelBKK020DF.write.saveAsTable("flooding_db.waterLvBKK0020")

if not tableCatalog.tableExists('flooding_db', 'waterLvBKK0021'):
    waterLevelBKK021DF.write.saveAsTable("flooding_db.waterLvBKK0021")

if not tableCatalog.tableExists('flooding_db', 'waterLVCY0015'):
    waterLevelCPY015DF.write.saveAsTable("flooding_db.waterLVCY0015")

print "Save 'Water Level' dataset of 8 stations before processing to HIVE successful"

```

Save Hourly Rainy Rate before processing to HIVE successful
Save 'Water Level' dataset of 8 stations before processing to HIVE successful

Took 22 sec. Last updated by anonymous at November 29 2020, 9:29:32 PM. (outdated)

1.4) ทดสอบดึงข้อมูลปริมาณน้ำฝนจาก HIVE โดยใช้ SQL

FINISHED

1.5) ทดสอบดึงข้อมูลระดับน้ำจาก HIVE โดยใช้ SQL

Took 0 sec. Last updated by anonymous at November 29 2020, 9:29:32 PM. (outdated)

1.4) Select 'Rainy Hourly Rate' from HIVE

SPARK JOBS FINISHED

```
%sql
select station_id, province_name_en, district, sub_district, date, H01, H02, H03, H04, H05,
       H06, H07, H08, H09, H10, H11, H12, H13, H14, H15, H16, H17, H18, H19, H20, H21, H22, H23
  , H24
from flooding_db.rainyHourly
where province_name_en = "Bangkok"
```

station_id	province_name_en	district	sub_district	d
4550001	Bangkok	Don Mueang	Sanam Bin	
4550001	Bangkok	Don Mueang	Sanam Bin	
4550001	Bangkok	Don Mueang	Sanam Bin	
4550001	Bangkok	Don Mueang	Sanam Bin	
4550001	Bangkok	Don Mueang	Sanam Bin	
4550001	Bangkok	Don Mueang	Sanam Bin	
4550001	Bangkok	Don Mueang	Sanam Bin	
4550001	Bangkok	Don Mueang	Sanam Bin	

Output is truncated to 102400 bytes. Learn more about ZEPPELIN_INTERPRETER_OUTPUT_LIMIT

Took 1 sec. Last updated by anonymous at November 29 2020, 9:29:34 PM. (outdated)

1.5) Select 'Water Level Rate' from HIVE

SPARK JOB (<http://hadoop-cluster-2020-msc-m.us-central1-f.c.abiding-orb-295915.internal:4040/jobs/job?id=32>) FINISHED

```
%sql
select * from flooding_db.waterLvBKK001
```

date	time	water_lv	station
2014-01-01 00:00:00.0	00:00:00	0.11	BKK001
2014-01-01 00:00:00.0	00:10:00	0.11	BKK001
2014-01-01 00:00:00.0	00:20:00	0.11	BKK001
2014-01-01 00:00:00.0	00:30:00	0.11	BKK001
2014-01-01 00:00:00.0	00:40:00	0.11	BKK001

flooding_project

Took 0 sec. Last updated by anonymous at November 29 2020, 9:29:34 PM. (outdated)

FINISHED

Step 2: Data Preprocessing

2.1) กรองข้อมูลในแหล่งแต่ข้อมูลที่ต้องการประมวลผล

จากข้อมูลปริมาณน้ำฝนจากพื้นที่หัวประเทศไทยนำกรองข้อมูลในแหล่งแต่ข้อมูลของกรุงเทพมหานคร โดยทำการ filter ด้วยเงื่อนไข

```
province_name_en = "Bangkok"
```

Took 0 sec. Last updated by anonymous at November 29 2020, 9:29:34 PM. (outdated)

2.1) Data Preprocessing - Filter only Bangkok

FINISHED

```
%pyspark
#filter only "Bangkok" province and assign to variable 'bangkokHourlyData'
bangkokHourlyData = hourlyData.filter(hourlyData['province_name_en']=="Bangkok").select(hourlyData['station_id'], hourlyData['province_name'],
hourlyData['sub_district'], hourlyData['date'], hourlyData['H01'], hourlyData['H02'], hourlyData['H03'], hourlyData['H04'], hourlyData[
H08'], hourlyData['H09'], hourlyData['H10'], hourlyData['H11'], hourlyData['H12'], hourlyData['H13'], hourlyData['H14'], hourlyData[
H18'], hourlyData['H19'], hourlyData['H20'], hourlyData['H21'], hourlyData['H22'], hourlyData['H23'], hourlyData['H24'])
print "filter only Bangkok for Hourly Rainy Rate"
```

filter only Bangkok for Hourly Rainy Rate

Took 0 sec. Last updated by anonymous at November 29 2020, 9:29:35 PM. (outdated)

2.2) และ 2.3) Replace missing value

FINISHED

จากข้อมูลข้อมูลปริมาณน้ำฝน กลุ่มผู้จัดทำเพิ่มข้อมูล NULL ในข้อมูลปริมาณน้ำฝนรายชั่วโมง ซึ่งข้อมูล NULL ได้กระจายตัวอยู่ตาม Column ชั่วโมงที่ 01:00 – 24:00 ดังนั้นกลุ่มผู้จัดทำจึงได้ทำการ Replace ข้อมูล NULL ด้วย “ค่าเฉลี่ยปริมาณน้ำฝนเฉพาะชั่วโมงที่ฝนตก” ในรันนั่น ตามทฤษฎี Statistical value of known data

```
ค่าเฉลี่ยปริมาณน้ำฝนเฉพาะชั่วโมงที่ฝนตก = SUM(ปริมาณน้ำฝนทั้งวัน) / COUNT(ชั่วโมงที่ฝนตก)
```

Took 0 sec. Last updated by anonymous at November 29 2020, 9:46:01 PM. (outdated)

2.2) Missing Value (Statistical value of known data); calculate average rainy rate of each that day

ESTATEKU (spark://spark01:7077, spark://spark02:7077, spark://spark03:7077, spark://spark04:7077) [job=4040/jobs/job?id=33] FINISHED

```
%pyspark
#calculate the number of rainy hours on that day. (rainy hour = 24 - dry hour)
import datetime

rainyHourArray = []
class RainyHour:
    def __init__(self, district, subDistrict, date, totalRainyHour, summaryRainyRate):
        self.key = str(abs(hash(district+subDistrict)))
        self.location = str(district + "/" + subDistrict)
        self.day = str(datetime.datetime.strptime(date, "%d/%m/%Y").date().day)
        self.month = str(datetime.datetime.strptime(date, "%d/%m/%Y").date().month)
        self.year = str(datetime.datetime.strptime(date, "%d/%m/%Y").date().year)
        self.totalRainyHour = totalRainyHour
        self.summaryRainyRate = summaryRainyRate
        self.averageRainyRate = self.calAvg()

    def info(self):
        return (self.key, self.location, self.day, self.month, self.year, self.totalRainyHour, self.summaryRainyRate, self.averageRainyRate)

    def calAvg(self):
        if self.totalRainyHour!=0 and self.summaryRainyRate>0.0:
            return round(self.summaryRainyRate / self.totalRainyHour, 2)
        else:
            return 0

    def calculateRainyHour(row):
        rainyHour=24
        summaryRainyRate = 0.0

        for i in range(1, 25):
            colName= "H0"+str(i) if i<=9 else "H"+str(i)

            if row[colName]!="NULL" and float(row[colName])==0.0:
                rainyHour-=1

            if row[colName]!="NULL" : summaryRainyRate+=float(row[colName])

        return RainyHour(row['district'], row['sub_district'], row['date'], rainyHour, summaryRainyRate)
```

```

for row in bangkokHourlyData.rdd.collect():
    r = calculateRainyHour(row)
    rainyHourArray.append(r.info())

def getAvgRainyRateByLocationAndDate(district, subDistrict, date):
    keyIn = str(abs(hash(district+subDistrict)))
    dayIn = str(datetime.datetime.strptime(date, "%d/%m/%Y").date().day)
    monthIn = str(datetime.datetime.strptime(date, "%d/%m/%Y").date().month)
    yearIn = str(datetime.datetime.strptime(date, "%d/%m/%Y").date().year)

    for r in rainyHourArray:
        if r[0]==keyIn and r[2]==dayIn and r[3]==monthIn and r[4]==yearIn:
            row = r[7]

    return row

print "Calculate average rainy rate of each that day. The result of this step will be used for missing value replacement on next step"
Calculate average rainy rate of each that day. The result of this step will be used for missing value replacement on next step

Took 2 sec. Last updated by anonymous at November 29 2020, 9:29:37 PM. (outdated)

```

2.3) Missing value (Statistical value of known data) - Replace NULL with average rainy rate of each that day

```

%pyspark
def replaceNullWithAvg(row):
    avgRainyRate = getAvgRainyRateByLocationAndDate(row['district'], row['sub_district'], row['date'])
    return (
        float(avgRainyRate if row['H01']=="NULL" else row['H01']),
        float(avgRainyRate if row['H02']=="NULL" else row['H02']),
        float(avgRainyRate if row['H03']=="NULL" else row['H03']),
        float(avgRainyRate if row['H04']=="NULL" else row['H04']),
        float(avgRainyRate if row['H05']=="NULL" else row['H05']),
        float(avgRainyRate if row['H06']=="NULL" else row['H06']),
        float(avgRainyRate if row['H07']=="NULL" else row['H07']),
        float(avgRainyRate if row['H08']=="NULL" else row['H08']),
        float(avgRainyRate if row['H09']=="NULL" else row['H09']),
        float(avgRainyRate if row['H10']=="NULL" else row['H10']),
        float(avgRainyRate if row['H11']=="NULL" else row['H11']),
        float(avgRainyRate if row['H12']=="NULL" else row['H12']),
        float(avgRainyRate if row['H13']=="NULL" else row['H13']),
        float(avgRainyRate if row['H14']=="NULL" else row['H14']),
        float(avgRainyRate if row['H15']=="NULL" else row['H15']),
        float(avgRainyRate if row['H16']=="NULL" else row['H16']),
        float(avgRainyRate if row['H17']=="NULL" else row['H17']),
        float(avgRainyRate if row['H18']=="NULL" else row['H18']),
        float(avgRainyRate if row['H19']=="NULL" else row['H19']),
        float(avgRainyRate if row['H20']=="NULL" else row['H20']),
        float(avgRainyRate if row['H21']=="NULL" else row['H21']),
        float(avgRainyRate if row['H22']=="NULL" else row['H22']),
        float(avgRainyRate if row['H23']=="NULL" else row['H23']),
        float(avgRainyRate if row['H24']=="NULL" else row['H24']))
    )

def extract(row):
    rowDate = row['date']
    day = str(datetime.datetime.strptime(rowDate, "%d/%m/%Y").date().day)
    month = str(datetime.datetime.strptime(rowDate, "%d/%m/%Y").date().month)
    year = str(datetime.datetime.strptime(rowDate, "%d/%m/%Y").date().year)

    return (
        row['station_id'], row['district'] + "/" + row['sub_district'], day, month, year, replaceNullWithAvg(row)
    )

cleanDF=bangkokHourlyData.rdd.map(extract).toDF(['StationId', 'Location','Day','Month', 'Year', 'RainyRates'])

print "Replace missing value by using average rainy rate of each that day successful"
Replace missing value by using average rainy rate of each that day successful

Took 2 sec. Last updated by anonymous at November 29 2020, 9:29:39 PM. (outdated)

```

2.4) Assert เพื่อตรวจสอบให้แน่ใจว่าไม่มีค่า NULL เกิดขึ้นแล้วในข้อมูลprimakanเน้าฝนรายชั่วโมง ตั้งแต่ Column ชั่วโมงที่ 01:00 till 24:00

Took 0 sec. Last updated by anonymous at November 29 2020, 9:29:39 PM. (outdated)

2.4) Missing value - Check rainy_rate since 01:00 till 24:00 should not be NULL

```

%pyspark
for row in cleanDF.rdd.collect():
    hourRow = row['RainyRates']
    for i in range(0, 23):
        assert "True" == str((hourRow[i] != "NULL"))

print "Assert not null to ensure that all of missing values were replaced successful"
Assert not null to ensure that all of missing values were replaced successful

Took 2 sec. Last updated by anonymous at November 29 2020, 9:29:42 PM. (outdated)

```

2.5) การแปลง Attribute

FINISHED

flooding-project67

การแปลง Date ที่อยู่ในรูปแบบเดือน-วัน-ปีเป็นชุด Date อยู่ในรูปแบบ “ปี-เดือน-วัน” ทางกลุ่มผู้จัดทำเห็นว่าหากไม่แปลง Attribute จะทำให้ยากต่อการประมวลผลข้อมูล จึงทำการแปลงข้อมูล Date ออกมาเป็น 3 Columns คือ “วัน”, “เดือน” และ “ปี”

เนื่องจากข้อมูลเดือนมีรูปแบบเป็นตัวเลข (Integer) ทำให้ไม่สามารถสื่อความหมายได้ดีนักเมื่อนำไปแสดงผลเป็นกราฟ ทางกลุ่มผู้จัดทำจึงทำการแปลงเดือนจากตัวเลข (Integer) เป็นตัวหนังสือ (String) โดยใช้ Arrays 12 ช่อง

Took 0 sec. Last updated by anonymous at November 29 2020, 9:29:43 PM. (outdated)

2.5) Preprocessing - Extract calendar column to day, month, year column on water-level dataset

SPARK JOBS FINISHED

```
%pyspark
import datetime
from pyspark.sql import Row

monthsArray = ["none", "January", "February", "March", "April", "May", "June", "July", "August", "September", "October", "November", "December"]

def extract(row):
    rowDate = str(row['date'])[:10]
    day = str(datetime.datetime.strptime(rowDate, "%Y-%m-%d").date().day)
    month = int(datetime.datetime.strptime(rowDate, "%Y-%m-%d").date().month)
    year = str(datetime.datetime.strptime(rowDate, "%Y-%m-%d").date().year)

    return (
        row['station'], day, month, monthsArray[(month)], year, row['time'], float(row['water_lv']))

waterLevelBKK001ExtractDF = waterLevelBKK001DF.rdd.map(extract).toDF(['Station', 'Day', 'Month', 'MonthName', 'Year', 'Time', 'WaterLevel'])
print "Separate calendar column of Water-Level of station BKK001 successful"
waterLevelBKK003ExtractDF = waterLevelBKK003DF.rdd.map(extract).toDF(['Station', 'Day', 'Month', 'MonthName', 'Year', 'Time', 'WaterLevel'])
print "Separate calendar column of Water-Level of station BKK003 successful"
waterLevelBKK004ExtractDF = waterLevelBKK004DF.rdd.map(extract).toDF(['Station', 'Day', 'Month', 'MonthName', 'Year', 'Time', 'WaterLevel'])
print "Separate calendar column of Water-Level of station BKK004 successful"
waterLevelBKK005ExtractDF = waterLevelBKK005DF.rdd.map(extract).toDF(['Station', 'Day', 'Month', 'MonthName', 'Year', 'Time', 'WaterLevel'])
print "Separate calendar column of Water-Level of station BKK005 successful"
waterLevelBKK008ExtractDF = waterLevelBKK008DF.rdd.map(extract).toDF(['Station', 'Day', 'Month', 'MonthName', 'Year', 'Time', 'WaterLevel'])
print "Separate calendar column of Water-Level of station BKK008 successful"
waterLevelBKK009ExtractDF = waterLevelBKK009DF.rdd.map(extract).toDF(['Station', 'Day', 'Month', 'MonthName', 'Year', 'Time', 'WaterLevel'])
print "Separate calendar column of Water-Level of station BKK009 successful"
waterLevelBKK020ExtractDF = waterLevelBKK020DF.rdd.map(extract).toDF(['Station', 'Day', 'Month', 'MonthName', 'Year', 'Time', 'WaterLevel'])
print "Separate calendar column of Water-Level of station BKK020 successful"
waterLevelBKK021ExtractDF = waterLevelBKK021DF.rdd.map(extract).toDF(['Station', 'Day', 'Month', 'MonthName', 'Year', 'Time', 'WaterLevel'])
print "Separate calendar column of Water-Level of station BKK021 successful"
waterLevelCPY015ExtractDF = waterLevelCPY015DF.rdd.map(extract).toDF(['Station', 'Day', 'Month', 'MonthName', 'Year', 'Time', 'WaterLevel'])
print "Separate calendar column of Water-Level of station CPY015 successful"

Separate calendar column of Water-Level of station BKK001 successful
Separate calendar column of Water-Level of station BKK003 successful
Separate calendar column of Water-Level of station BKK004 successful
Separate calendar column of Water-Level of station BKK005 successful
Separate calendar column of Water-Level of station BKK008 successful
Separate calendar column of Water-Level of station BKK009 successful
Separate calendar column of Water-Level of station BKK020 successful
Separate calendar column of Water-Level of station BKK021 successful
Separate calendar column of Water-Level of station CPY015 successful
```

Took 1 sec. Last updated by anonymous at November 29 2020, 9:29:44 PM. (outdated)

2.6) Replace abnormal value

FINISHED

จากข้อมูลระดับน้ำจาก 9 สถานีรอดพบร่วมค่าระดับน้ำที่ผิดปกติอยู่คือ -999 ซึ่งกลุ่มผู้จัดได้ Replace ด้วยค่า 0.0

```
ระดับน้ำ > 0.0 : ระดับน้ำในแหล่งน้ำ สูงกว่า ระดับน้ำทะเล
ระดับน้ำ = 0.0 : ระดับน้ำในแหล่งน้ำ เท่ากับ ระดับน้ำทะเล
ระดับน้ำ < 0.0 : ระดับน้ำในแหล่งน้ำ ต่ำกว่า ระดับน้ำทะเล
```

Took 0 sec. Last updated by anonymous at November 29 2020, 9:29:45 PM. (outdated)

2.6) Cleansing Data - Replease abnormal value -999 of water-level

SPARK JOBS FINISHED

```
%pyspark
from pyspark.sql import Row

def replaseAbnormalWaterLevel(row):
    waterLv = row['WaterLevel']
    if float(row['WaterLevel'])==float(-999):
        waterLv = 0.0
    return (
        row['Station'], row['Day'], row['Month'], row['MonthName'], row['Year'], row['Time'], waterLv
    )

waterLevelBKK001CleanDF = waterLevelBKK001ExtractDF.rdd.map(replaseAbnormalWaterLevel).toDF(['Station', 'Day', 'Month', 'MonthName', 'Year',
print "Replace abnormal value in Water-Level of station BKK001 successful"
```

```
waterLevelBKK003CleanDF = waterLevelBKK003ExtractDF.rdd.map(replaceAbnormalWaterLevel).toDF(['Station', 'Day', 'Month', 'MonthName', 'Year', ''])
print "Replace abnormal value in Water-Level of station BKK003 successful"
waterLevelBKK004CleanDF = waterLevelBKK004ExtractDF.rdd.map(replaceAbnormalWaterLevel).toDF(['Station', 'Day', 'Month', 'MonthName', 'Year', ''])
print "Replace abnormal value in Water-Level of station BKK004 successful"
waterLevelBKK005CleanDF = waterLevelBKK005ExtractDF.rdd.map(replaceAbnormalWaterLevel).toDF(['Station', 'Day', 'Month', 'MonthName', 'Year', ''])
print "Replace abnormal value in Water-Level of station BKK005 successful"
waterLevelBKK008CleanDF = waterLevelBKK008ExtractDF.rdd.map(replaceAbnormalWaterLevel).toDF(['Station', 'Day', 'Month', 'MonthName', 'Year', ''])
print "Replace abnormal value in Water-Level of station BKK008 successful"
waterLevelBKK009CleanDF = waterLevelBKK009ExtractDF.rdd.map(replaceAbnormalWaterLevel).toDF(['Station', 'Day', 'Month', 'MonthName', 'Year', ''])
print "Replace abnormal value in Water-Level of station BKK009 successful"
waterLevelBKK020CleanDF = waterLevelBKK020ExtractDF.rdd.map(replaceAbnormalWaterLevel).toDF(['Station', 'Day', 'Month', 'MonthName', 'Year', ''])
print "Replace abnormal value in Water-Level of station BKK020 successful"
waterLevelBKK021CleanDF = waterLevelBKK021ExtractDF.rdd.map(replaceAbnormalWaterLevel).toDF(['Station', 'Day', 'Month', 'MonthName', 'Year', ''])
print "Replace abnormal value in Water-Level of station BKK021 successful"
waterLevelCPY015CleanDF = waterLevelCPY015ExtractDF.rdd.map(replaceAbnormalWaterLevel).toDF(['Station', 'Day', 'Month', 'MonthName', 'Year', ''])
print "Replace abnormal value in Water-Level of station CPT015 successful"
```

```
Replace abnormal value in Water-Level of station BKK001 successful
Replace abnormal value in Water-Level of station BKK003 successful
Replace abnormal value in Water-Level of station BKK004 successful
Replace abnormal value in Water-Level of station BKK005 successful
Replace abnormal value in Water-Level of station BKK008 successful
Replace abnormal value in Water-Level of station BKK009 successful
Replace abnormal value in Water-Level of station BKK020 successful
Replace abnormal value in Water-Level of station BKK021 successful
Replace abnormal value in Water-Level of station CPT015 successful
```

Took 5 sec. Last updated by anonymous at November 29 2020, 9:29:50 PM. (outdated)

Step 3: Mapping

FINISHED

3.1) Map ข้อมูลจาก ผลรวมปริมาณน้ำฝน และค่าเฉลี่ยน้ำฝนของแต่ละวัน

RainyRate: A Ray Tracer for Monte-Carlo Rain Simulation

Bayan Alykate. Kulluktaan kutsutut tietäjät (Tietäjät ja tietäjien mukaan) 2023-08-29 10:59 PM (EST)

```
%pyspark
```

```

monthsArray = ["none", "January", "February", "March", "April", "May", "June", "July", "August", "September", "October", "November", "December"]

def summaryRainyRate(row):
    summaryRate = 0.0
    hourRow = row['RainyRates']
    result = [ row['StationId'], row['Location'], row['Day'], str(row['Month']), monthsArray[int(row['Month'])], row['Year'], row['RainyRate']]
    for i in range(0, 23):
        summaryRate+=hourRow[i]
    result.append(round(float(summaryRate), 2))
    return result

summaryDF = cleanDF.rdd.map(summaryRainyRate).toDF(["StationId", "Location", "Day", "MonthNo", "MonthName", "Year", "RainyRate", "DayRainyRate"])
summaryDF.show()

```

TOOK 4 SEC. Last updated by anonymous at November 29 2020, 9:29:54 PM. (Outdated)

Step 4: Reducing

FINISHED

Phase #1: Reduce ข้อมูลปริมาณน้ำฝนรายวัน และ ข้อมูลระดับน้ำรายวัน ในทักษะเป็นข้อมูลรายเดือน

(input) ข้อมูลปริมาณน้ำฝนรายวัน => Reduce => (output) ข้อมูลปริมาณน้ำฝนรายเดือน

flooding projectâ€ž

(input) ข้อมูลระดับน้ำรายเดือน => Reduce => (output) ข้อมูลระดับน้ำรายเดือน

Phase #2: นำ outputs จาก Reduce Phase #1 มา integrate กัน และทำการ Reduce อีกครั้งเพื่อได้ข้อมูลสรุปปริมาณน้ำฝนเทียบกับระดับน้ำของแต่ละพื้นที่

(input) ข้อมูลปริมาณน้ำฝนรายเดือน, ข้อมูลระดับน้ำรายเดือน => Reduce => (output) ข้อมูลสรุปปริมาณน้ำฝนเทียบกับระดับน้ำของแต่ละพื้นที่

Took 0 sec. Last updated by anonymous at November 29 2020, 9:29:54 PM. (outdated)

4.1) Reduce ข้อมูลปริมาณน้ำฝน โดยการหา Summary ด้วยเงื่อนไข รหัสสถานี, พื้นที่, เดือน และปี ดัง code ต่อไปนี้

FINISHED

```
.groupBy(["StationId", "Location", "MonthName", "MonthNo", "Year"])
```

Output จากการ summary จะถูกเก็บลงใน column sum(DayRainyRate)

Took 0 sec. Last updated by anonymous at November 29 2020, 9:29:54 PM. (outdated)

4.1) Reduce - Summary rainy rate by month and year

SPARK JOBS FINISHED

```
%pyspark  
groupByMYDF = summaryDF.groupBy(["StationId", "Location", "MonthName", "MonthNo", "Year"]).sum("DayRainyRate")  
z.show(groupByMYDF)  
groupByMYDF.cache()
```

grid chart pie line scatter settings ▾

StationId	Location	MonthName	MonthNo	Year
4550019	Lat Krabang/Khlong Sam Prawet	February	2	2014
4550025	Bang Khun Thian/Samae Dam	January	1	2014
4550004	Bang Khen/Tha Raeng	May	5	2014
4550018	Saphan Sung/Saphan Sung	May	5	2014
4550016	Lat Krabang/Lam Pla Thio	March	3	2014
4550004	Bang Khen/Tha Raeng	January	1	2014
4550026	Bang Khun Thian/Tha Kham	March	3	2014
4550026	Bang Khun Thian/Tha Kham	April	4	2014
4550027	Khlong Toei/Khlong Toei	July	7	2014
4550014	Nong Chok/Krathum Rai	January	1	2014
4550001	Don Mueang/Sanam Bin	March	3	2014
4550018	Saphan Sung/Saphan Sung	April	4	2014
4550024	Bang Bon/Bang Bon	April	4	2014

DataFrame[StationId: bigint, Location: string, MonthName: string, MonthNo: string, Year: string, sum(DayRainyRate): double]

Took 13 sec. Last updated by anonymous at November 29 2020, 9:30:07 PM. (outdated)

4.2) Reduce ข้อมูลระดับน้ำจากทั้ง 9 สถานีโดยการหา ค่าเฉลี่ย (mean) ของระดับน้ำในแต่ละเดือน

FINISHED

(input) ข้อมูลระดับน้ำจากทั้ง 9 สถานี => Reduce => (output) ข้อมูลค่าเฉลี่ยระดับน้ำในแต่ละเดือนทั้งหมด 9 สถานี

ทำการ union ข้อมูลระดับน้ำเฉลี่ยในแต่ละเดือนของทั้ง 9 สถานี รวมเป็น 1 dataset

(input) ข้อมูลระดับน้ำเฉลี่ยในแต่ละเดือนของทั้ง 9 สถานี => Reduce => ข้อมูลระดับน้ำเฉลี่ยในแต่ละเดือนของสถานีทั้งหมดในพื้นที่ กรุงเทพฯ

Output:

- Station: รหัสสถานี

- 1: ค่าเฉลี่ยระดับน้ำในเดือนมกราคม
- 2: ค่าเฉลี่ยระดับน้ำในเดือนกุมภาพันธ์

flooding_projectâ€ž

- 12: ค่าเฉลี่ยระดับน้ำในเดือนธันวาคม

Took 0 sec. Last updated by anonymous at November 29 2020, 9:30:07 PM. (outdated)

4.2) Reduce - Find average of water-level dataset from 9 stations and union them all to one finally dataset.

SPARK JOBS FINISHED

```
%pyspark
def reduceCleanWaterLevelData(waterlvDF):
    return waterlvDF.filter(waterlvDF['Year'] == "2014").groupBy(["Station"]).pivot("Month").mean("WaterLevel")

waterLevelBKK001SummaryDF = reduceCleanWaterLevelData(waterLevelBKK001CleanDF)
print "Station 'BKK001' was completely reduced"

waterLevelBKK003SummaryDF = reduceCleanWaterLevelData(waterLevelBKK003CleanDF)
print "Station 'BKK003' was completely reduced"

waterLevelBKK004SummaryDF = reduceCleanWaterLevelData(waterLevelBKK004CleanDF)
print "Station 'BKK004' was completely reduced"

waterLevelBKK005SummaryDF = reduceCleanWaterLevelData(waterLevelBKK005CleanDF)
print "Station 'BKK005' was completely reduced"

waterLevelBKK008SummaryDF = reduceCleanWaterLevelData(waterLevelBKK008CleanDF)
print "Station 'BKK008' was completely reduced"

waterLevelBKK009SummaryDF = reduceCleanWaterLevelData(waterLevelBKK009CleanDF)
print "Station 'BKK009' was completely reduced"

waterLevelBKK020SummaryDF = reduceCleanWaterLevelData(waterLevelBKK020CleanDF)
print "Station 'BKK020' was completely reduced"

waterLevelBKK021SummaryDF = reduceCleanWaterLevelData(waterLevelBKK021CleanDF)
print "Station 'BKK021' was completely reduced"

waterLevelCPY015SummaryDF = reduceCleanWaterLevelData(waterLevelCPY015CleanDF)
print "Station 'CPY015' was completely reduced"

waterLevelFinalDF = waterLevelBKK001SummaryDF.union(waterLevelBKK003SummaryDF).union(waterLevelBKK004SummaryDF).union(waterLevelBKK005SummaryDF).union(waterLevelBKK008SummaryDF).union(waterLevelBKK020SummaryDF).union(waterLevelBKK021SummaryDF).union(waterLevelCPY015SummaryDF)
z.show(waterLevelFinalDF)

Station 'BKK001' was completely reduced
Station 'BKK003' was completely reduced
Station 'BKK004' was completely reduced
Station 'BKK005' was completely reduced
Station 'BKK008' was completely reduced
Station 'BKK009' was completely reduced
Station 'BKK020' was completely reduced
Station 'BKK021' was completely reduced
Station 'CPY015' was completely reduced
```

Station	1	2	3	4	5	
BKK001	-0.033736559139785 224	-0.094890873015872 66	-0.067181899641576 14	-0.121002314814813 37	-0.100056003584229 3	-4.8 E-4
BKK003	0.7388037634408569	0.7790749007936503	0.8023745519713245	0.6482222222222206	0.5563821684587792	0.39 6
BKK004	0.5165412186379946	0.5463070436507858	0.5184923835125449	0.4258680555555585	0.3643862007168473	0.23 4
BKK005	0.2557974910394269 5	0.2699975198412655	0.254348118279568	0.1880324074074083	0.1551164874551973	0.08 8
BKK008	0.0533221326164879 1	0.0252802579365078 8	0.0042921146953404 25	-0.0976597222222222 1	-0.176823476702511 28	-0.1 5

Took 4 min 39 sec. Last updated by anonymous at November 29 2020, 9:34:46 PM. (outdated)

4.3) นำผลลัพธ์การ Reduce จากข้อ 4.1 และ 4.2 มา integrate กันด้วย relationship ระหว่าง 2 fields ดังนี้

FINISHED

Field	Description	Dataset	Example Data

Field	Description	Dataset	Example Data
StationId	รหัสสถานีตรวจวัดปริมาณน้ำฝน	ข้อมูลปริมาณน้ำฝนจากชั้นตอนที่ 4.1	4550019, 4550025, 4550008
StationCode	รหัสสถานีตรวจวัดระดับน้ำ	ข้อมูลค่าเฉลี่ยระดับน้ำจากชั้นตอนที่ 4.2	BKK001, BKK004, BKK021, CPY015

โดยทางกลุ่มได้นำข้อมูลเพื่อจับคู่ว่า สถานีตรวจวัดปริมาณน้ำฝน โดยอยู่ในพื้นที่เดียวกันกับ สถานีตรวจวัดระดับน้ำ และทำเป็น Table Mapping ไว้ดังนี้

รหัสสถานีตรวจวัดระดับน้ำ	ชื่อสถานีตรวจวัดระดับน้ำ	รหัสสถานีตรวจวัดปริมาณน้ำฝนที่อยู่ในพื้นที่
BKK001	คลองลาดพร้าว ท้ายปต.คลอง2	4550004, 4550005, 4550006, 4550020
BKK004	คลองทวีวัฒนา ท้ายปต.ทวีวัฒนา	4550007
BKK005	คลองภาษีเจริญ เพชรเกษม69	4550021, 4550024, 4550025, 4550026
BKK008	คลองแส้นแสบ บางกะปี	4550003, 4550017, 4550018, 4550022, 4550027
BKK009	คลองลำปลาทิว ลาดกระบัง	4550002, 4550009, 4550012, 4550013, 4550014, 4550016, 4550019, 4550023
BKK021	คลองลาดพร้าว วัดบางบัว	4550001, 4550008, 4550010, 4550011
CPY015	สะพานกรุงเทพ	4550015, 4550028

Took 0 sec. Last updated by anonymous at November 29 2020, 9:34:46 PM. (outdated)

เมื่อนำมา plot กราฟด้วย แกน X = ปริมาณน้ำฝน, แกน Y = ระดับน้ำจากสถานี โดยจัดกลุ่มตามพื้นที่ จะได้กราฟดังนี้

FINISHED

จะสังเกตได้ว่าข้อมูลที่ plot ในกราฟนั้นคุยกันเนื่องจาก ลักษณะข้อมูลที่มีความแตกต่างกันมากเกินไป

- ข้อมูลปริมาณน้ำฝนมีช่วงข้อมูลอยู่ที่ 0.0 – 200.00
- ข้อมูลระดับน้ำช่วงตั้งแต่ -2.00 – 2.00

Took 0 sec. Last updated by anonymous at November 29 2020, 9:34:47 PM. (outdated)

4.3) Reduce - 2 Sources integration. Rainy rate data & Water-level data

SPARK JOBS FINISHED

```
%pyspark
import pyspark.sql.functions as F
from pyspark.sql.types import StringType

bkk001Station = waterLevelBKK001SummaryDF.first()
bkk004Station = waterLevelBKK004SummaryDF.first()
bkk005Station = waterLevelBKK005SummaryDF.first()
bkk008Station = waterLevelBKK008SummaryDF.first()
bkk009Station = waterLevelBKK009SummaryDF.first()
bkk021Station = waterLevelBKK021SummaryDF.first()
cpy015Station = waterLevelCPY015SummaryDF.first()

stationDict = {
    "BKK001": ["4550004", "4550005", "4550006", "4550020"],
    "BKK004": ["4550007"],
    "BKK005": ["4550021", "4550024", "4550025", "4550026"],
    "BKK008": ["4550003", "4550017", "4550018", "4550022", "4550027"],
    "BKK009": ["4550002", "4550009", "4550012", "4550013", "4550014", "4550016", "4550019", "4550023"],
    "BKK021": ["4550001", "4550008", "4550010", "4550011"],
    "CPY015": ["4550015", "4550028"]
}

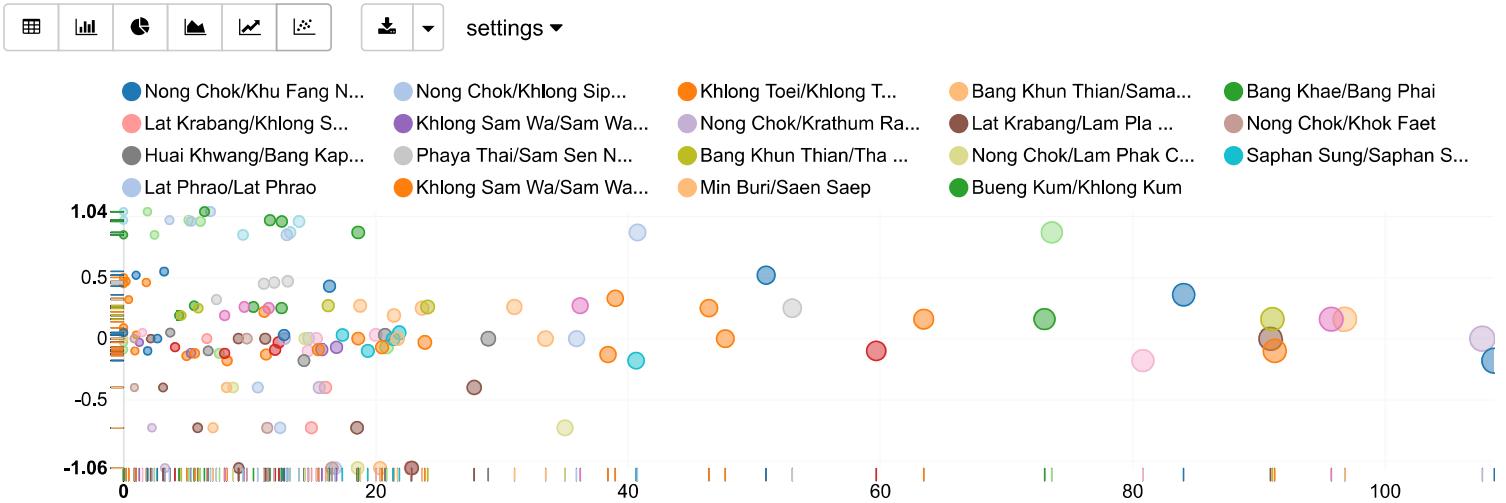
def reduceDF(row):
    stationID = str(row['StationId'])
    stationCode = "BKK001"
    waterLv= 0.0
    if stationID in stationDict["BKK001"]:
        stationCode = "BKK001"
        waterLv = bkk001Station[row['MonthNo']]
    elif stationID in stationDict["BKK004"]:
        stationCode = "BKK004"
        waterLv = bkk004Station[row['MonthNo']]
    elif stationID in stationDict["BKK005"]:
        stationCode = "BKK005"
        waterLv = bkk005Station[row['MonthNo']]
    elif stationID in stationDict["BKK008"]:
        stationCode = "BKK008"
        waterLv = bkk008Station[row['MonthNo']]
    elif stationID in stationDict["BKK009"]:
        stationCode = "BKK009"
```

```

waterLv = bkk009Station[row['MonthNo']]
elif stationID in stationDict["BKK021"]:
    stationCode = "BKK021"
    waterLv = bkk021Station[row['MonthNo']]
elif stationID in stationDict["CPY015"]:
    stationCode = "CPY015"
    waterLv = cpy015Station[row['MonthNo']]
else:
    print "Station Id: {} not in configuration".format(stationID)
return (
    row['Location'], row['MonthName'], round(row['sum(DayRainyRate)'],2), round(waterLv,2), row['Location']+ " ("+row['MonthName']+")"
)

```

integratedDF = groupByMYDF.rdd.map(reduceDF).toDF(['Location', 'Month', 'MonthlyRainyRate', 'MonthlyWaterLevel', 'LocationAndMonth']).sort('z').show(integratedDF)



Took 2 min 10 sec. Last updated by anonymous at November 29 2020, 9:36:57 PM. (outdated)

การทำ Standardization ข้อมูล “ปริมาณน้ำฝน” และ “ระดับน้ำ” เพื่อนำข้อมูลทั้ง 2 อยู่ในช่วงเดียวกันสามารถเปรียบเทียบกันได้ โดยใช้ MinMaxScaler FINISHED

หลักจากทำ Standardization ด้วย MinMaxScaler พบว่าเมื่อนำข้อมูลไป plot กราฟแล้วง่ายต่อความเข้าใจมากขึ้น โดยสามารถสรุปการวิเคราะห์ข้อมูลได้ดังนี้ ดังนี้

Pattern	Problem	Locations	Graph's Position
ปริมาณน้ำฝนสูง ระดับน้ำสูง	พื้นที่ไม่มีปัญหา แม้ฝนตกหนักแต่ยังสามารถระบายน้ำลงไปสู่แหล่งน้ำได้ดี	<ul style="list-style-type: none"> เขตคลองเตย/แขวงคลองเตย (ในบางเดือน) เขตหนองจอก/แขวงคุ้งฝังเหนือ เขตบางขุนเทียน/แขวงแสมดำและท่าข้าม เขตลาดพร้าว/แขวงลาดพร้าว เขตบางบอน/บางบอน เขตสัมพันธวงศ์/แขวงทุ่งสองห้อง อื่นๆ 	มุมขวาบนของกราฟ
ปริมาณน้ำฝนสูง ระดับน้ำต่ำ	พื้นที่มีปัญหา เนื่องจากไม่สามารถระบายน้ำปริมาณน้ำฝนลงไปยังแหล่งน้ำได้ อาจเกิดจากปัญหาท่อระบายน้ำตัน	<ul style="list-style-type: none"> เขตคลองสามวา/แขวงสามวาตะวันตก เขตหนองจอก / แขวงกระทุมราย เขตลาดกระบัง / แขวงล่าปลาทิว เขตหนองจอก / แขวงโคลแฟด อื่นๆ 	มุมขวาล่างของกราฟ

Pattern	Problem	Locations	Graph's Position
ปริมาณน้ำฝนต่ำ ระดับน้ำสูง	พื้นที่ผู้ประสบภัย เนื่องจากฝนตกน้อย แต่ระดับน้ำสูง ปั่นบกอกถึงปั๊มจั้ยอื่น เช่นภาวะน้ำทะลุนูนหรือน้ำเหนือ	<ul style="list-style-type: none"> เขตปีงกุ่ม / แขวงคลองกุ่ม เขตลาดกระบัง / แขวงคลองสามัคคี เขตคลองเตย/แขวงคลองเตย (ในบางเดือน) อื่นๆ 	มุมซ้ายล่างของกราฟ
ปริมาณน้ำฝนต่ำ ระดับน้ำต่ำ	พื้นที่ไม่มีปัญหา เนื่องจากฝนตกน้อย และ ระดับน้ำต่ำ	<ul style="list-style-type: none"> เขตบางขุนเทียน / แขวงท่าข้าม (ในบางเดือน) เขตสะพานสูง / แขวงสะพานสูง เขตบางบอน / แขวงบางบอน เขตหนองจอก / แขวงคลองสิบสอง เขตบางแค / แขวงบางไผ อื่นๆ 	มุมซ้ายล่างของกราฟ

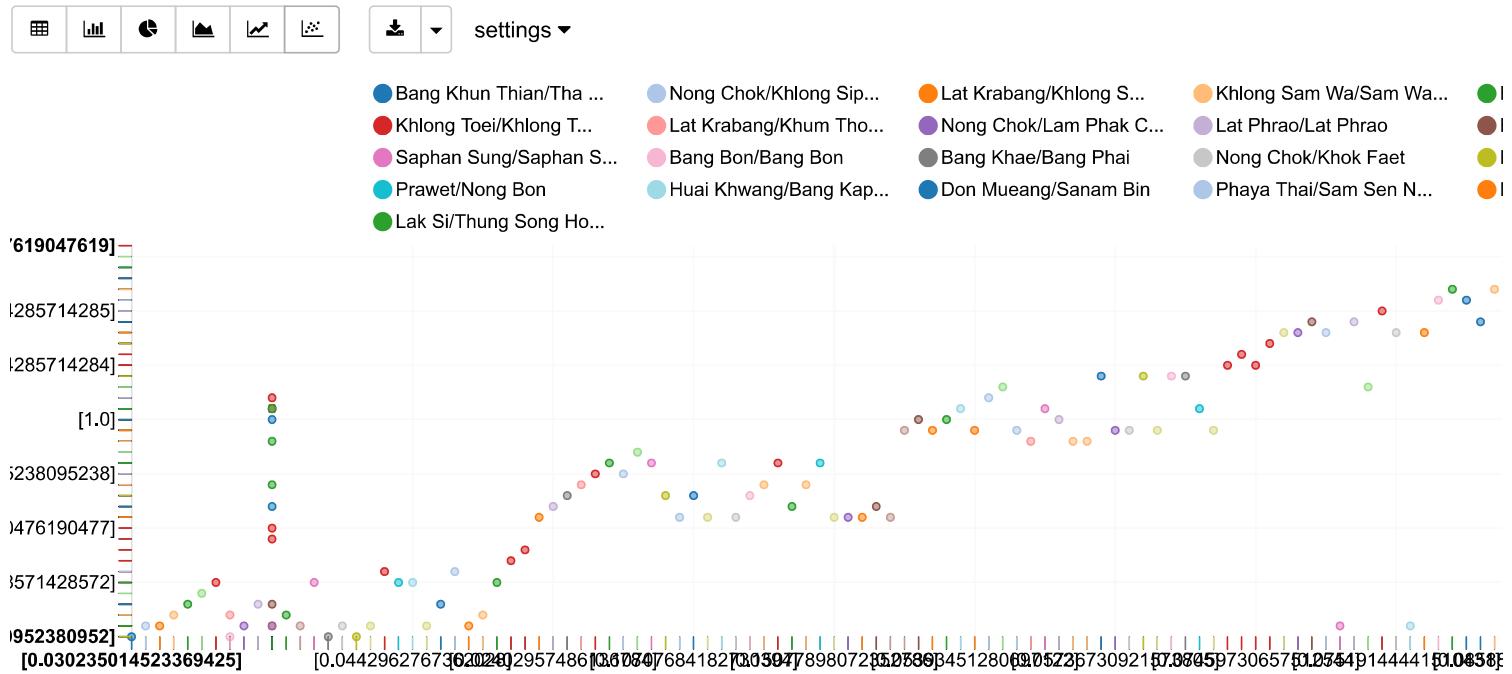
Took 0 sec. Last updated by anonymous at November 29 2020, 9:36:57 PM. (outdated)

4.3) Standardization MonthlyRainyRate and MonthlyWaterLevel by MinMaxScaler

SPARK JOBS FINISHED

```
%pyspark
from pyspark.ml import Pipeline
from pyspark.ml.feature import MinMaxScaler
from pyspark.ml.feature import VectorAssembler

targetColumnsScale = ["MonthlyRainyRate", "MonthlyWaterLevel"]
assemblers = [VectorAssembler(inputCols=[col], outputCol=col + "Vector") for col in targetColumnsScale]
scalers = [MinMaxScaler(inputCol=col + "Vector", outputCol=col + "Scaled") for col in targetColumnsScale]
pipeline = Pipeline(stages=assemblers+scalers)
scalerModel = pipeline.fit(integratedDF)
scaledRainyRateAndWaterLvDF = scalerModel.transform(integratedDF)
z.show(scaledRainyRateAndWaterLvDF)
```



Took 21 sec. Last updated by anonymous at November 29 2020, 9:37:18 PM. (outdated)

4.4) Reduce เพื่อหาค่าเสียงเกิดปัญหาน้ำท่วมของแต่ละพื้นที่ โดยใช้สมการดังนี้

FINISHED

ค่าความเสี่ยง = ปริมาณน้ำฝน / ระดับน้ำ

หากผลลัพธ์ยิ่งมีค่าสูง แสดงว่าพื้นที่นั้น ยิ่งมีความเสี่ยงที่จะเกิดปัญหาน้ำท่วมสูงนั้นเอง

Took 0 sec. Last updated by anonymous at November 29 2020, 9:37:18 PM. (outdated)

flooding_projectâ€ž

อ้างอิงข้อมูลจาก:

FINISHED

กรมอุตุนิยมวิทยา. หนังสืออุตุนิยมวิทยา ฤดูกาลของประเทศไทย [ออนไลน์]. แหล่งที่มา: <https://www.tmd.go.th/info/info.php?FileID=53> [19 พฤศจิกายน 2563]

Took 0 sec. Last updated by anonymous at November 29 2020, 9:37:18 PM. (outdated)

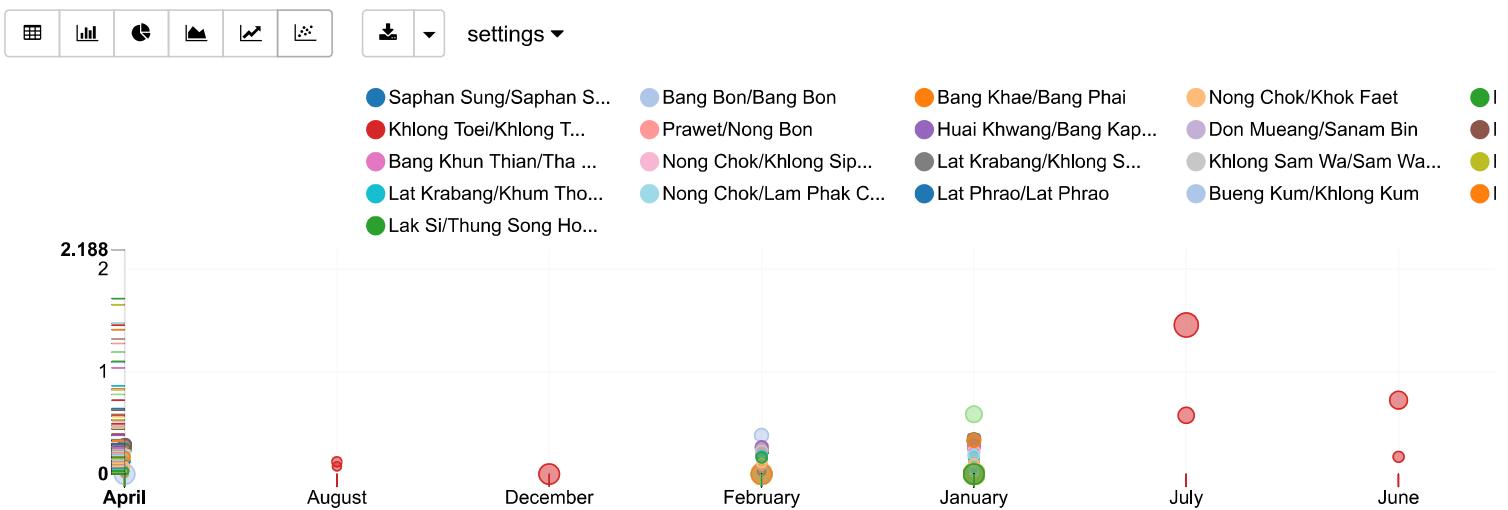
4.4) Reduce - Finding critical area on each month

SPARK JOBS FINISHED

%pyspark

```
def rainyRateDivideWaterLv(row):
    rainyRate = row['MonthlyRainyRateScaled']
    waterLevel = row['MonthlyWaterLevelScaled']
    criticalScore = rainyRate/waterLevel
    criticalScore = round(float(criticalScore[0]), 4)
    if criticalScore == float('Infinity'):
        criticalScore = 0.0000
    return (
        row['Location'], row['Month'], criticalScore
    )
```

```
criticalScoreDF = scaledRainyRateAndWaterLvDF.rdd.map(rainyRateDivideWaterLv).toDF(['Location', 'Month', 'CriticalScore'])
z.show(criticalScoreDF)
```



Took 5 sec. Last updated by anonymous at November 29 2020, 9:37:24 PM. (outdated)

4.5) บันทึกข้อมูลหลังจากการประมวลผลใน HIVE

SPARK JOBS FINISHED

```
%pyspark
tableCatalog = spark._jsparkSession.catalog()
if not tableCatalog.tableExists('flooding_db', 'criticalScore'):
    criticalScoreDF.write.saveAsTable("flooding_db.criticalScore")

print "บันทึกลงใน HIVE ตารางที่ชื่อว่า flooding_db.criticalScore สำเร็จเรียบร้อย"
print "บันทึกลงใน HIVE ตารางที่ชื่อว่า flooding_db.criticalScore สำเร็จเรียบร้อย"
```

Took 2 sec. Last updated by anonymous at November 29 2020, 9:37:27 PM. (outdated)

Step 5: Clustering

FINISHED

5.1) จัดกลุ่มข้อมูลโดยการแบ่ง Clustering ด้วยวิธี K-means และทำการ validate จำนวน Clustering ด้วย WCSSE และ Elbow

จากการลองจัดกลุ่มข้อมูลด้วย K=2 จนถึง K=20 และหา WCSSE เพื่อ plot กราฟ

พบว่าความชันของกราฟคงที่ตั้งแต่ K=20 ไปลงมาจนถึง K=6 และ ความชันเริ่มเปลี่ยนแปลงตั้งแต่ K=5 เป็นต้นไป

ดังนั้นสรุปได้ว่าจำนวน Clusters ที่เหมาะสมที่สุดคือ 5 Clusters

Took 0 sec. Last updated by anonymous at November 29 2020, 9:37:27 PM. (outdated)

5.1) Clustering by using K-means & Validate Clustering by using WCSSE

SPARK JOBS FINISHED

```
%pyspark
from pyspark.ml.feature import VectorAssembler
from pyspark.ml.clustering import KMeans
from pyspark.ml.feature import StandardScaler

vecAssembler = VectorAssembler(inputCols=["CriticalScore"], outputCol="features")
sourceDF = vecAssembler.transform(criticalScoreDF)

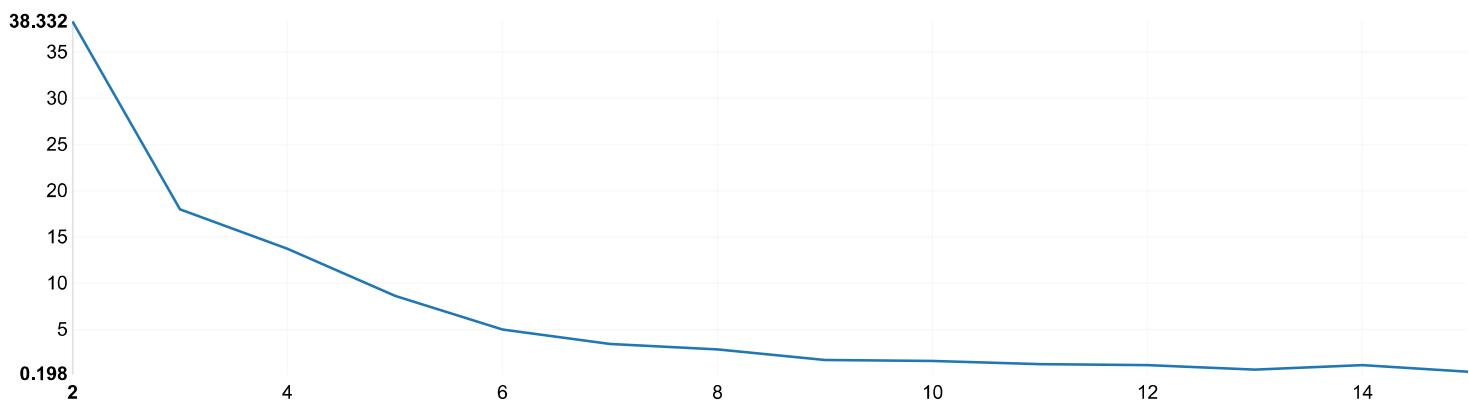
scaler = StandardScaler(inputCol="features", outputCol="scaledFeatures", withStd=True, withMean=False)
scalerModel = scaler.fit(sourceDF)
finalSourcedata = scalerModel.transform(sourceDF)

def getWCSSE(k):
    k+=1
    row = []
    for i in range(2, k):
        kmeansI = KMeans(featuresCol='scaledFeatures',k=i)
        modelKI = kmeansI.fit(finalSourcedata)
        wcsseKI = modelKI.computeCost(finalSourcedata)
        row.append((i, wcsseKI))
    resultWcsseDF = spark.createDataFrame(row, ['k', 'WCSSE'])
    return resultWcsseDF

resultWcsseDF = getWCSSE(20)

z.show(resultWcsseDF)
```

grid chart pie line area download settings ▾



Took 55 sec. Last updated by anonymous at November 29 2020, 9:38:22 PM. (outdated)

Step 6: Outlier Detection

FINISHED

6.1) หา Outlier จากข้อมูล ผลกระทบปริมาณน้ำฝนรายเดือนแบ่งตามพื้นที่พบรwa

ผลกระทบปริมาณน้ำฝน > 46.56 = Upper Bound

ผลกระทบปริมาณน้ำฝน < -19.68 = Lower Bound

6.2) หา Outlier จากข้อมูล ค่าเฉลี่ยระดับน้ำรายเดือนแบ่งตามพื้นที่พบรwa

ผลกระทบปริมาณน้ำฝน > 0.819548917131 = Upper Bound

ผลกระทบปริมาณน้ำฝน < -0.350834284283 = Lower Bound

Took 0 sec. Last updated by anonymous at November 29 2020, 9:38:23 PM. (outdated)

6.1) Outlier Detection - Rainy Rate

SPARK JOBS FINISHED

```
%pyspark
import pyspark.sql.functions as F
from pyspark.sql.types import StringType

summaryDF = groupByMYDF.select("sum(DayRainyRate)").summary("min", "max", "25%", "75%", "mean", "stddev")
min = float(summaryDF.collect()[0][1])
max = float(summaryDF.collect()[1][1])
q1 = float(summaryDF.collect()[2][1])
q3 = float(summaryDF.collect()[3][1])
mean = float(summaryDF.collect()[4][1])
sd = float(summaryDF.collect()[5][1])
summaryDF.show()
```

```

iqr = q3-q1
upperB = q3+(1.5*iqr)
lowerB = q1-(1.5*iqr)
print "IQR: {}".format(iqr)
print "Upper Bound: {} / Lower Bound: {}".format(upperB, lowerB)

def checkBoundary(sumRate):
    if sumRate > upperB:
        return 'upper bound'
    elif sumRate < lowerB:
        return 'lower bound'
    else :
        return 'normal'

checkBoundaryFunc = F.udf(checkBoundary, StringType())

```

```

OutlierDF = groupByMYDF.withColumn("Outlier", checkBoundaryFunc("sum(DayRainyRate)"))
z.show(OutlierDF)

```

```

+-----+
|summary| sum(DayRainyRate)|
+-----+
|   min|      0.0|
|   max|  151.48|
|  25%|      5.16|
| 75%|21.72000000000002|
| mean|21.69893333333333|
| stddev| 28.49236933398102|
+-----+

```

IQR: 16.56
 Upper Bound: 46.56 / Lower Bound: -19.68

settings ▾

StationId	Location	MonthName	MonthNo	Year	
4550018	Saphan Sung/Saphan Sung	May	5	2014	40.
4550018	Saphan Sung/Saphan Sung	April	4	2014	19.
4550018	Saphan Sung/Saphan Sung	February	2	2014	17.
4550018	Saphan Sung/Saphan Sung	March	3	2014	21.
4550018	Saphan Sung/Saphan Sung	January	1	2014	21.

Took 7 sec. Last updated by anonymous at November 29 2020, 9:38:30 PM. (outdated)

6.2) Outlier Detection - Water level

SPARK JOBS FINISHED

```

%pyspark
from pyspark.sql.functions import col

inYearDF = waterLevelFinalDF.select("Station", (((col("1") + col("2") + col("3") + col("4") + col("5") + col("6") + col("7") + col("8") + col("9") + col("10")))
inYearStatisticDF=inYearDF.select("inYearAverage").summary("min", "max", "25%", "75%", "mean", "stddev")
minInYear = float(inYearStatisticDF.collect()[0][1])
maxInYear = float(inYearStatisticDF.collect()[1][1])
q1InYear = float(inYearStatisticDF.collect()[2][1])
q3InYear = float(inYearStatisticDF.collect()[3][1])
meanInYear = float(inYearStatisticDF.collect()[4][1])
sdInYear = float(inYearStatisticDF.collect()[5][1])
inYearStatisticDF.show()

iqrInYear = q3InYear-q1InYear
upperBInYear = q3InYear+(1.5*iqrInYear)
lowerBInYear = q1InYear-(1.5*iqrInYear)
print "IQR: {}".format(iqrInYear)
print "Upper Bound: {} / Lower Bound: {}".format(upperBInYear, lowerBInYear)

def checkBoundaryInYear(inYear):
    if inYear > upperB:
        return 'upper bound'
    elif inYear < lowerB:
        return 'lower bound'
    else :
        return 'normal'

checkBoundaryFunc = F.udf(checkBoundaryInYear, StringType())
outlierInYearDF = inYearDF.withColumn("Outlier", checkBoundaryFunc("inYearAverage"))
outlierInYearDF.show()

```

```
+-----+-----+
|summary|      inYearAverage|
+-----+-----+
| min| -0.21126636949209213|
| 25%|  0.08805941624708387|
| 75%|  0.38065521660052815|
| mean|  0.30638183549919973|
| stddev|  0.37306760760637947|
+-----+-----+
```

IQR: 0.292595800353
Upper Bound: 0.819548917131 / Lower Bound: -0.350834284283

```
+-----+-----+
|Station|      inYearAverage|Outlier|
+-----+-----+
| BKK001|  0.08805941624708387| normal|
| BKK002|  0.602611870185961| normal|
+-----+-----+
```

Took 3 min 45 sec. Last updated by anonymous at November 29 2020, 9:42:15 PM. (outdated)

FINISHED

งานวิจัยที่เกี่ยวข้อง

จากแผนปฏิบัติการป้องกันและแก้ไขปัญหาน้ำท่วมกรุงเทพมหานคร พบว่าสาเหตุของน้ำท่วมเกิดได้จากการธรรมชาติ เช่น น้ำฝน, น้ำทุ่ง, น้ำเนื้อ, น้ำทะเล น Hun และระดับน้ำในแม่น้ำเจ้าพระยา และสเหตุทางกายภาพได้แก่ ปัญหาผังเมือง, ปัญหาการระบายน้ำ และปัญหาแผ่นดินไหว

จากหัวข้อ 7.2 แผนป้องกันน้ำท่วมเนื่องจากฝนตกหนัก ภารกิจที่สำนักการระบายน้ำได้ทำมากที่สุดในปีได้แก่

- การตรวจสอบเครื่องสูบน้ำ
- การดำเนินการเปิดทางน้ำให้ล่องคล่อง
- การทำความสะอาดท่อระบายน้ำ

ชี้ง 3 กิจกรรมดังกล่าวจะทำติดต่อกันเป็นระยะเวลาทั้งหมด 8 เดือน (มี.ค. - ต.ค.)

ดังนั้นหากมีการน้ำข้อมูลสถิติปริมาณน้ำฝนและระดับน้ำมาทำการวิเคราะห์แบ่งแยกตามพื้นที่และแยกตามช่วงเวลา เพื่อประกอบการตัดสินใจวางแผนแก้ไขปัญหาน้ำท่วม จะส่งผลให้กระบวนการแก้ปัญหาน้ำท่วมเกิดประสิทธิภาพสูงสุด

อ้างอิง:

สำนักการระบายน้ำ. (2561). แผนปฏิบัติการป้องกันและแก้ไขปัญหาน้ำท่วมกรุงเทพมหานคร เนื่องจากน้ำฝนและน้ำ Hun ประจำปี 2561.
[ออนไลน์]. แหล่งที่มา: http://203.155.220.119/News_dds/magazine/Plan61/05.pdf
(http://203.155.220.119/News_dds/magazine/Plan61/05.pdf)

Took 0 sec. Last updated by anonymous at November 29 2020, 9:42:15 PM. (outdated)