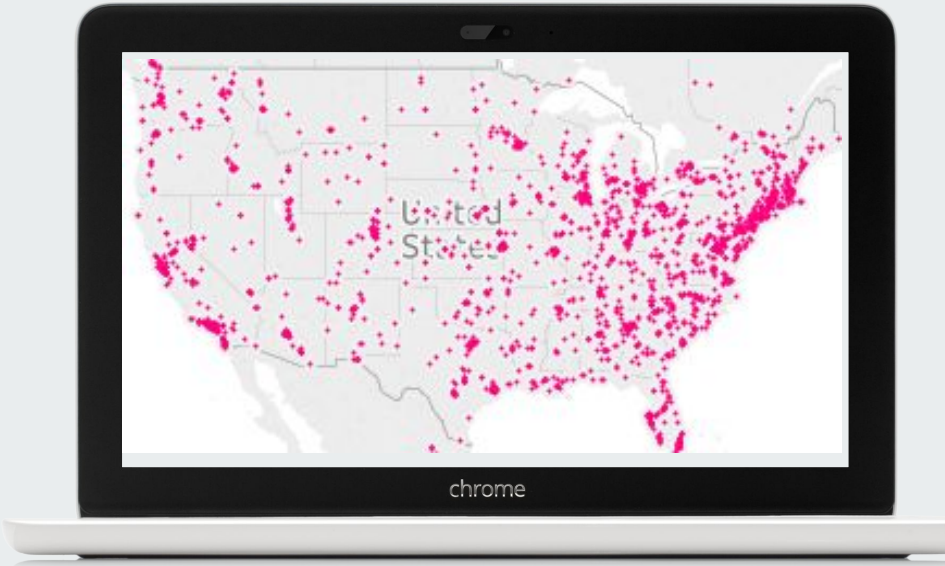
Big Data Project

Mohamed Selem
Shang Gao

2018.3.19 10:00 am



Outline

Project 1: Twitter stream analysis

Project 2: Crime data analysis

Project 3: Text clustering

Conclusion

Project 1

Twitter stream analysis

https://github.com/MuhammadElsayed/twitter_stream_kafka

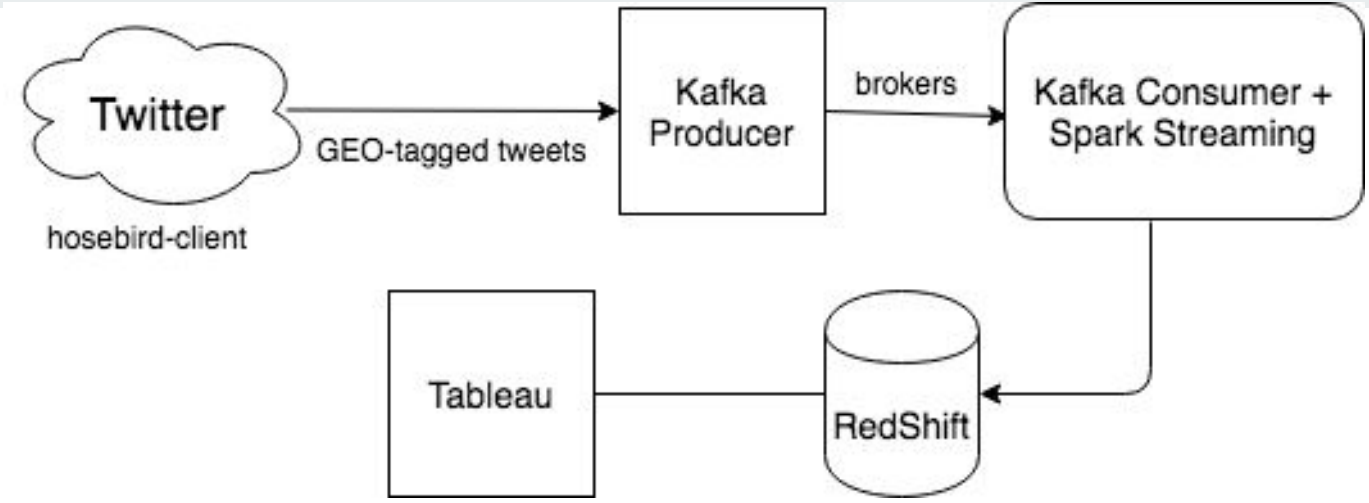
https://us-east-1.online.tableau.com/#/site/shanggao/workbooks/265129/views

Story

- Fetch twitter stream data
- Analyze real-time stream data
- Find something interesting based on geolocation and topic

Technologies

Architecture



- src/main/java
  - db
    - RedShiftDataEmitter.java
  - kafka
    - KafkaConstants.java
    - TwitterKafkaConsumer.java
    - TwitterKafkaConsumerWithSpark.java
    - TwitterKafkaProducer.java
    - TwitterSourceConstant.java
  - model
    - Tweet.java

```
        <dependency>
            <groupId>org.apache.kafka</groupId>
            <artifactId>kafka_2.11</artifactId>
            <version>0.8.2.1</version>
        </dependency>
        <dependency>
            <groupId>org.apache.spark</groupId>
            <artifactId>spark-streaming_2.11</artifactId>
            <version>2.2.0</version>
        </dependency>
        <dependency>
            <groupId>org.apache.spark</groupId>
            <artifactId>spark-streaming-kafka-0-10_2.11</artifactId>
            <version>2.2.0</version>
        </dependency>
```

Kafka (Producer/Consumer)

- In the producer, we are fetching the tweets into a Linked Blocking Queue

Linked => Order Blocking Queue => thread safe

- Filtering the valid tweets (has a real coordinations) inside the producer.
- We used only one broker.
- Serializing the data using Kafka String Serializer and the same for deserialization.

Spark Streaming

- After couple of tries, We have configured the patch period to 2 seconds for the streaming context (generated average 60 records/RDD).
- Used direct streaming approach rather than receiver-based approach as it’s newer, more efficient, and easier to understand, and it makes a one-to-one mapping between partitions in a Kafka topic and partitions in a Spark RDD.

Spark Streaming

```
[JobScheduler] INFO org.apache.spark.streaming.scheduler.JobScheduler - Starting job streaming job 1521445382000 ms.0 from job set of time 1521445
[JobGenerator] INFO org.apache.spark.streaming.scheduler.JobScheduler - Added jobs for time 1521445382000 ms
[streaming-job-executor-0] INFO org.apache.spark.SparkContext - Starting job: foreach at TwitterKafkaConsumerWithSpark.java:52
[dag-scheduler-event-loop] INFO org.apache.spark.scheduler.DAGScheduler - Got job 253 (foreach at TwitterKafkaConsumerWithSpark.java:52) with 1 ou
[dag-scheduler-event-loop] INFO org.apache.spark.scheduler.DAGScheduler - Final stage: ResultStage 253 (foreach at TwitterKafkaConsumerWithSpark.j
[dag-scheduler-event-loop] INFO org.apache.spark.scheduler.DAGScheduler - Parents of final stage: List()
[dag-scheduler-event-loop] INFO org.apache.spark.scheduler.DAGScheduler - Missing parents: List()
[dag-scheduler-event-loop] INFO org.apache.spark.scheduler.DAGScheduler - Submitting ResultStage 253 (KafkaRDD[253] at createDirectStream at Twitt
[dag-scheduler-event-loop] INFO org.apache.spark.storage.memory.MemoryStore - Block broadcast_253 stored as values in memory (estimated size 3.0 K
[dag-scheduler-event-loop] INFO org.apache.spark.storage.memory.MemoryStore - Block broadcast_253_piece0 stored as bytes in memory (estimated size
[dispatcher-event-loop-1] INFO org.apache.spark.storage.BlockManagerInfo - Added broadcast_253_piece0 in memory on 10.0.2.15:34945 (size: 1962.0 B
[dag-scheduler-event-loop] INFO org.apache.spark.SparkContext - Created broadcast 253 from broadcast at DAGScheduler.scala:1006
[dag-scheduler-event-loop] INFO org.apache.spark.scheduler.DAGScheduler - Submitting 1 missing tasks from ResultStage 253 (KafkaRDD[253] at create
[dag-scheduler-event-loop] INFO org.apache.spark.scheduler.TaskSchedulerImpl - Adding task set 253.0 with 1 tasks
[dispatcher-event-loop-0] INFO org.apache.spark.scheduler.TaskSetManager - Starting task 0.0 in stage 253.0 (TID 253, localhost, executor driver,
[Executor task launch worker for task 253] INFO org.apache.spark.executor.Executor - Running task 0.0 in stage 253.0 (TID 253)
[Executor task launch worker for task 253] INFO org.apache.spark.streaming.kafka010.KafkaRDD - Computing topic twitter, partition 0 offsets 118321
[Executor task launch worker for task 253] INFO org.apache.spark.executor.Executor - Finished task 0.0 in stage 253.0 (TID 253). 751 bytes result
[task-result-getter-1] INFO org.apache.spark.scheduler.TaskSetManager - Finished task 0.0 in stage 253.0 (TID 253) in 286 ms on localhost (executo
[task-result-getter-1] INFO org.apache.spark.scheduler.TaskSchedulerImpl - Removed TaskSet 253.0, whose tasks have all completed, from pool
[dag-scheduler-event-loop] INFO org.apache.spark.scheduler.DAGScheduler - ResultStage 253 (foreach at TwitterKafkaConsumerWithSpark.java:52) finis
[streaming-job-executor-0] INFO org.apache.spark.scheduler.DAGScheduler - Job 253 finished: foreach at TwitterKafkaConsumerWithSpark.java:52, took
[JobScheduler] INFO org.apache.spark.streaming.scheduler.JobScheduler - Finished job streaming job 1521445382000 ms.0 from job set of time 1521445
[JobScheduler] INFO org.apache.spark.streaming.scheduler.JobScheduler - Total delay: 0.345 s for time 1521445382000 ms (execution: 0.337 s)
[JobGenerator] INFO org.apache.spark.streaming.kafka010.KafkaRDD - Removing RDD 252 from persistence list
[dispatcher-event-loop-0] INFO org.apache.spark.storage.BlockManagerInfo - Removed broadcast_251_piece0 on 10.0.2.15:34945 in memory (size: 1962.0
[block-manager-slave-async-thread-pool-6] INFO org.apache.spark.storage.BlockManager - Removing RDD 252
[JobGenerator] INFO org.apache.spark.streaming.scheduler.ReceivedBlockTracker - Deleting batches:
[JobGenerator] INFO org.apache.spark.streaming.scheduler.InputInfoTracker - remove old batch metadata: 1521445378000 ms
[dispatcher-event-loop-0] INFO org.apache.spark.storage.BlockManagerInfo - Removed broadcast_252_piece0 on 10.0.2.15:34945 in memory (size: 1962.0
--- New RDD with 1 partitions and 54 records
```

Why Amazon RedShift?

- Solving “dark data” problem (tweets data mostly has this problem)
- Super-fast local disk performance.
- Sophisticated query optimization.
- Join-optimized data formats.
- Query using standard SQL.
- Optimized for data warehousing.

Amazon RedShift

Cluster: **bdt-twitter-project**

Configuration | Status | Cluster Performance | Queries | Loads | Table restore

Endpoint bdt-twitter-project.cgsgl8tibau7.us-east-2.redshift.amazonaws.com:5439 (authorized)

Cluster Properties

Cluster Name	bdt-twitter-project
Cluster Type	Single Node
Node Type	dc2.large
Nodes	1
Zone	us-east-2c
Created Time	March 16, 2018 at 3:50:26 PM UTC-5
Cluster Version	1.0.1885
VPC ID	vpc-d474b7bc (View VPCs)
Cluster Subnet Group	default
VPC security groups	MyWebDMZ (sg-02c2d06a) (active)
Cluster Parameter Group	default.redshift-1.0 (in-sync)
Enhanced VPC Routing	No

Cluster Status

Cluster Status	available
Database Health	healthy
In Maintenance Mode	no
Parameter Group Apply Status	in-sync
Pending Modified Values	None

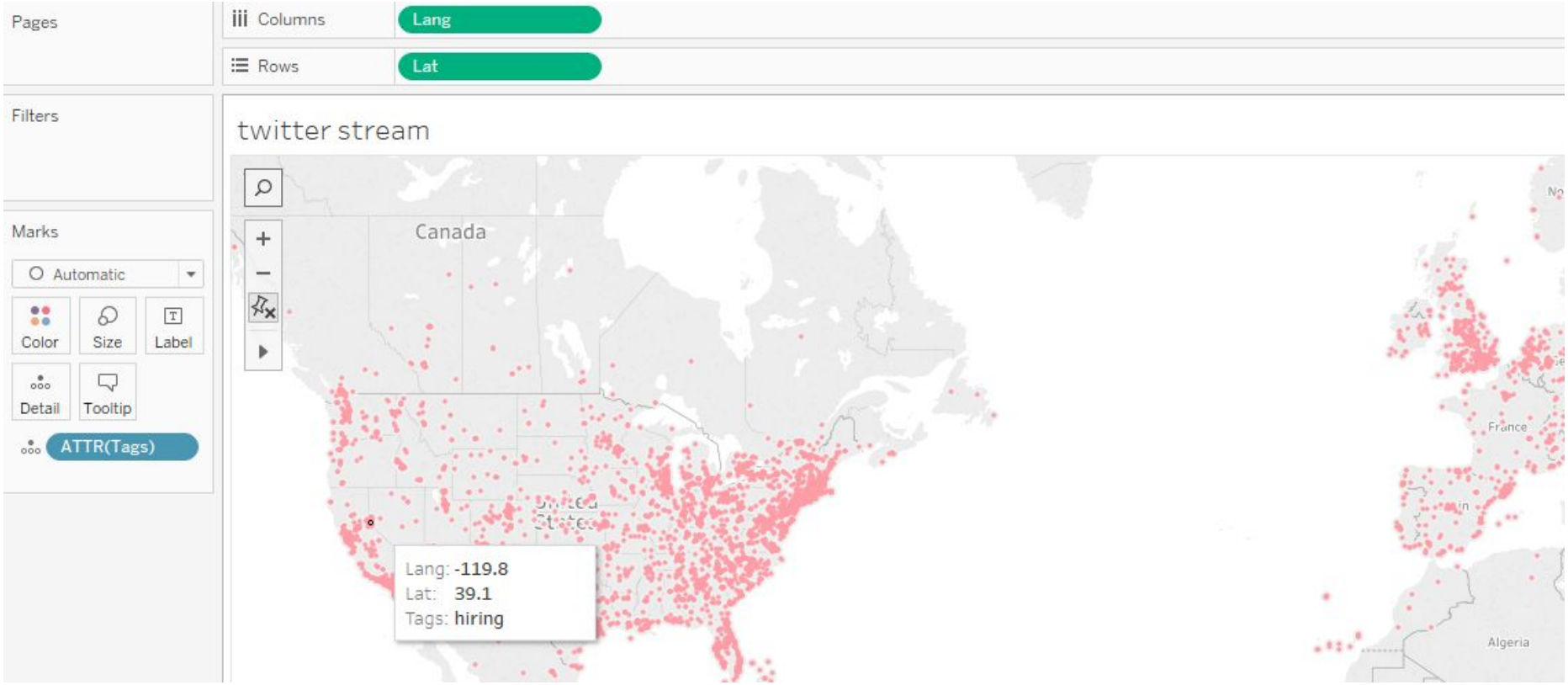
Cluster Database Properties

Port	5439
Publicly Accessible	Yes
Database Name	twitts
Master Username	bdt_twitter_user
Encrypted	No

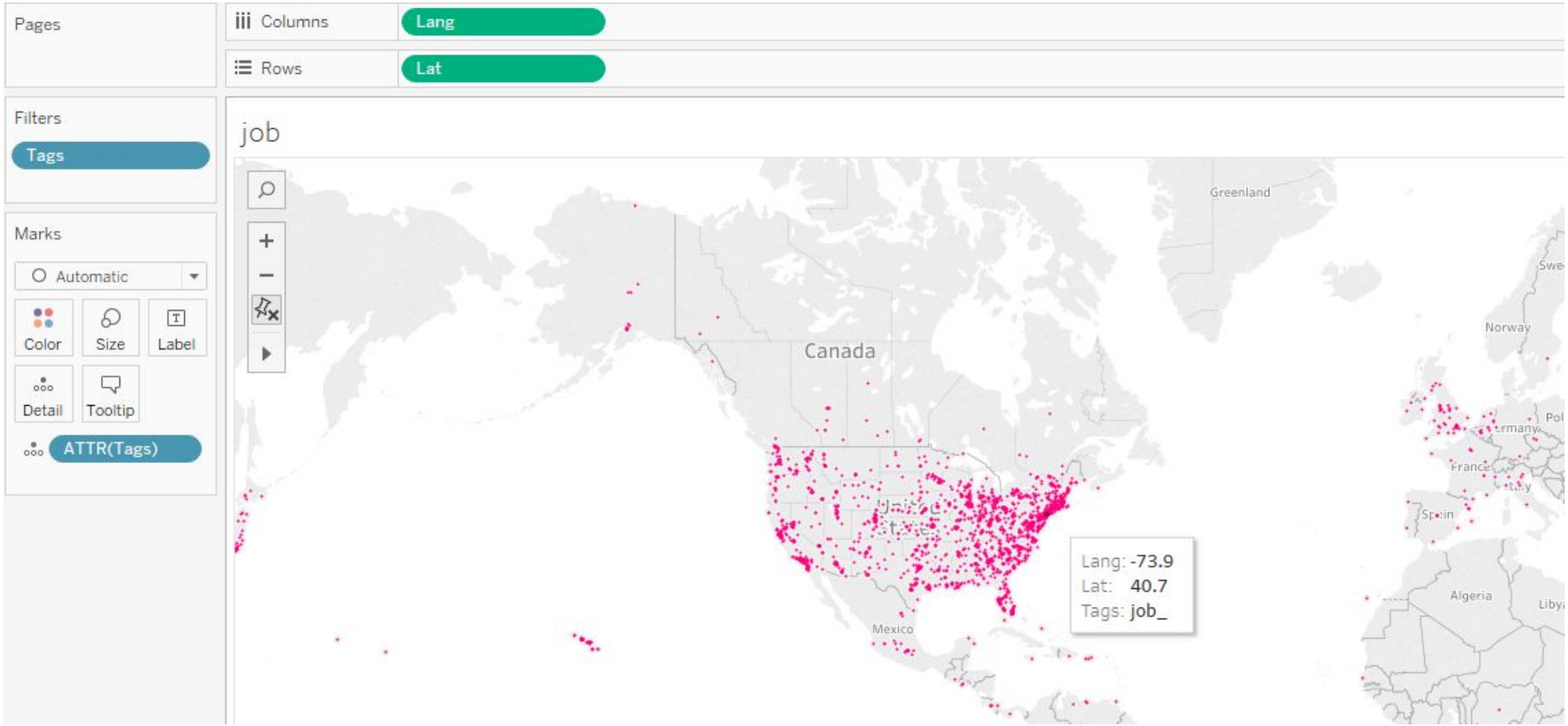
Backup, Audit Logging, and Maintenance

Automated Snapshot Retention Period	1
Cross-Region Snapshots Enabled	No
Audit Logging Enabled	No
Maintenance Window	fri:08:00-fri:08:30
Allow Version Upgrade	Yes

Visualization- all tags



Visualization- job tag



Project 2

Crime data analysis

https://community.cloud.databricks.com/?o=6289929760174924#notebook/4352377016250780/command/109189020959857 2

https://us-east-1.online.tableau.com/#/site/shanggao/workbooks/265098/views

Story

- Police Department Incidents Dataset(San Francisco)
- 15 years incidents data, 2003-2017
- more than 2 millions rows, 440 MB
- https://data.sfgov.org

Technologies

Spark SQL

databricks

+ableau

TABLEAU ONLINE

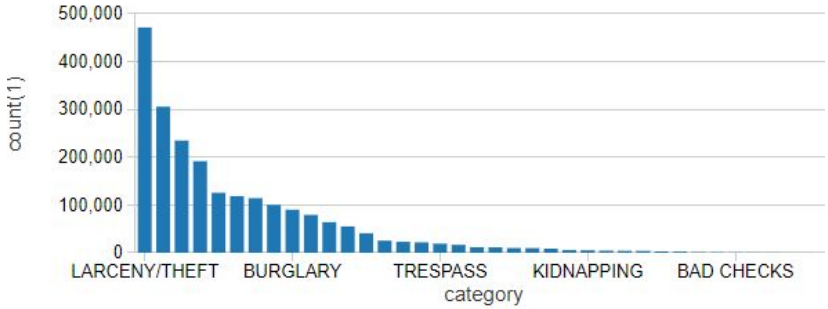
```
--IncidntNum,Category,Descript,DayOfWeek,Date,Time,PdDistrict,Resolution,Address,X,Y,Location,PdId
create table pdi (
        incidntnum  String,
        category  String,
        descript   String,
        dayofweek   String,
        date_   String,
        time_  String,
        pddistrict  String,
        resolution  String,
        address  String,
        x DECIMAL(30,20),
        y DECIMAL(30,20),
        location String,
        pdid String
)
COMMENT 'Police Department Incidents'
ROW FORMAT DELIMITED
FIELDS TERMINATED BY ','
LINES TERMINATED BY '\n'
STORED AS TEXTFILE
;

LOAD DATA LOCAL INPATH '/home/cloudera/SF_Police_Department_Incidents.csv' OVERWRITE INTO TABLE pdi;
```

SparkSQL-1

select category, count(*) from pdi group by category order by 2 desc

```
1  select   category, count(*) from pdi group by category order by 2 desc
```

▸ (1) Spark Jobs

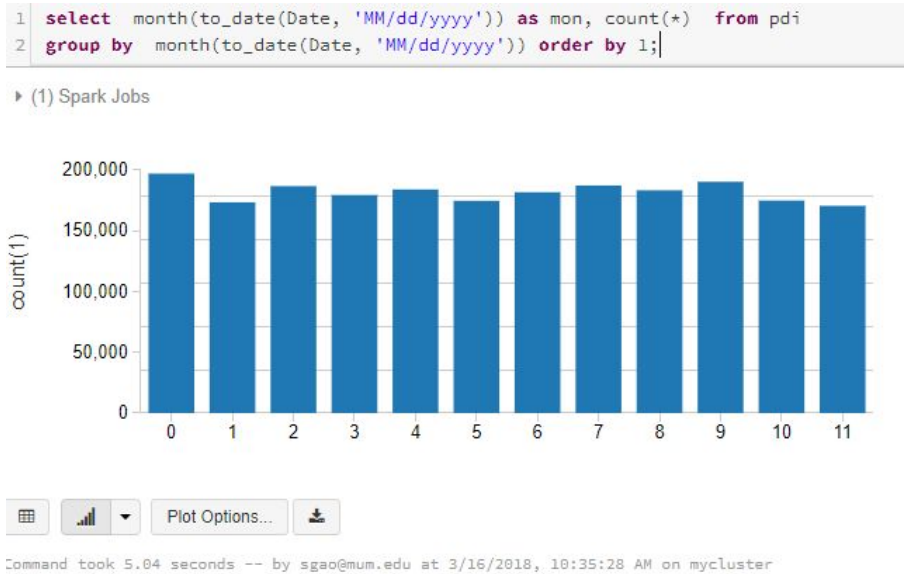


Plot Options...

Command took 4.29 seconds -- by sgao@mum.edu at 3/16/2018, 11:11:11 AM on mycluster

SparkSQL-2

```
select  split(Time, ':')[0]
, count(*)  from pdi
group by split(Time,
':')[0] order by 1;
```

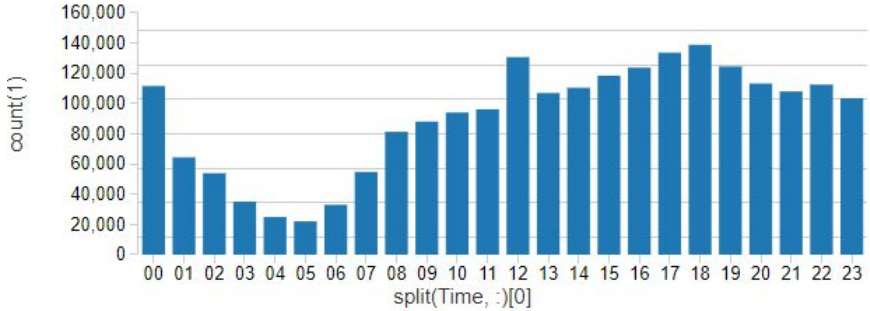


SparkSQL-3

select category, count(*) from pdi group by category order by 2 desc

```
1 select  split(Time, ':')[0]  , count(*)  from pdi
2 group by split(Time, ':')[0] order by 1;
```

▸ (1) Spark Jobs



Plot Options...

Command took 6.07 seconds -- by sgao@mum.edu at 3/16/2018, 11:26:25 AM on mycluster

Comparison between Hive & Spark

#	description	Hive on vm sql	Spark on hive vm pyspark	Spark sql on databrick
1	count(*)	278.8	23.081	3.26
2	Count by category	255.927	31.637	4.29
3	Count by month	129.801	17.332	5.04
4	Count by hours	125.998	17.878	4.75
5	Count by each ten minutes	133.13	18.892	4.54
6	Count by week	185.08	41.460	5.13
7	Self inner join	152.717	50.684	6.55

Tableau-1

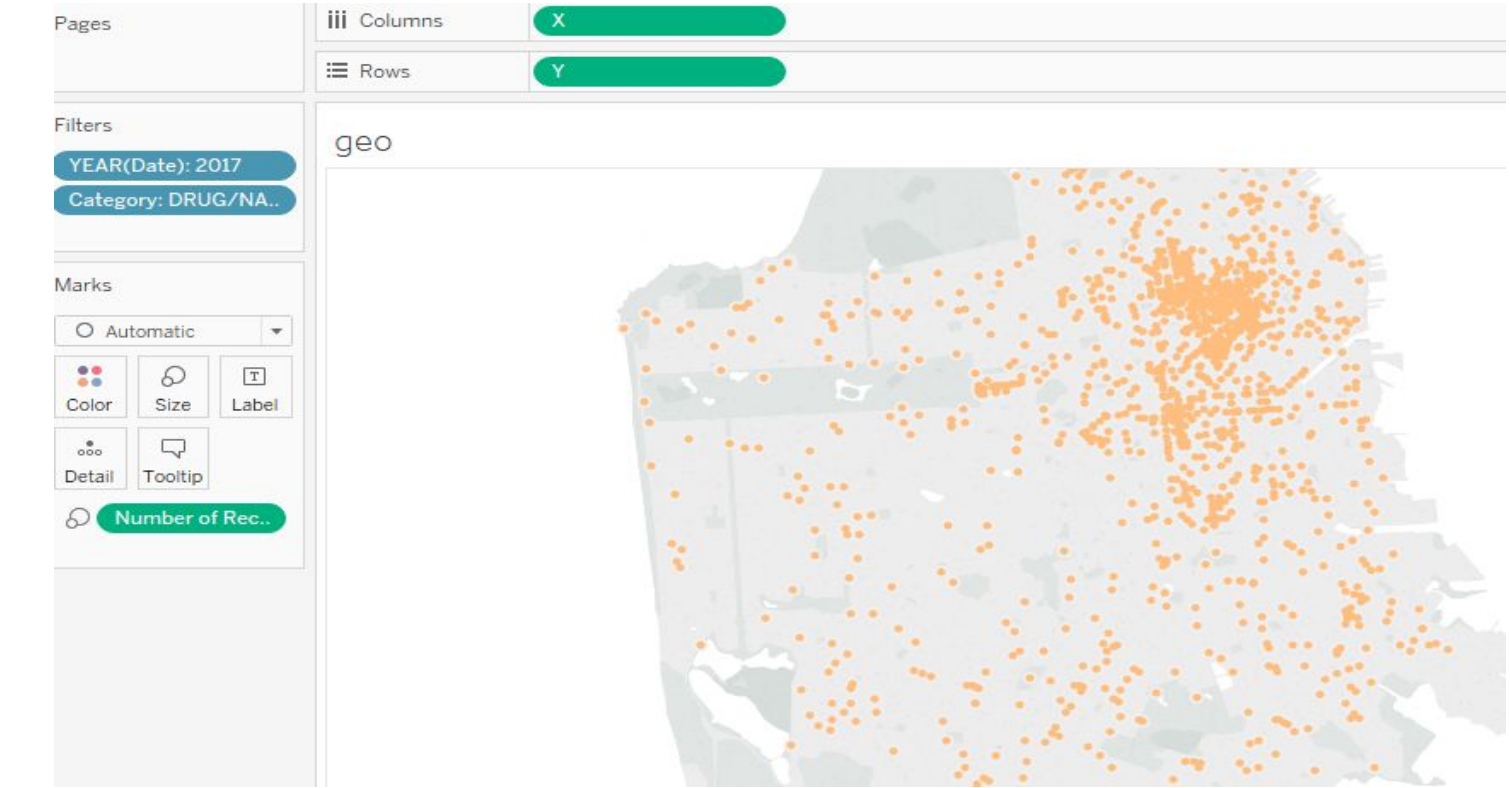

Pages
Columns
X
Rows
Y
Filters
YEAR(Date): 2017
Category: DRUG/NA..
Marks
Automatic
Color
Size
Label
Detail
Tooltip
Number of Rec..
geo

Tableau-2

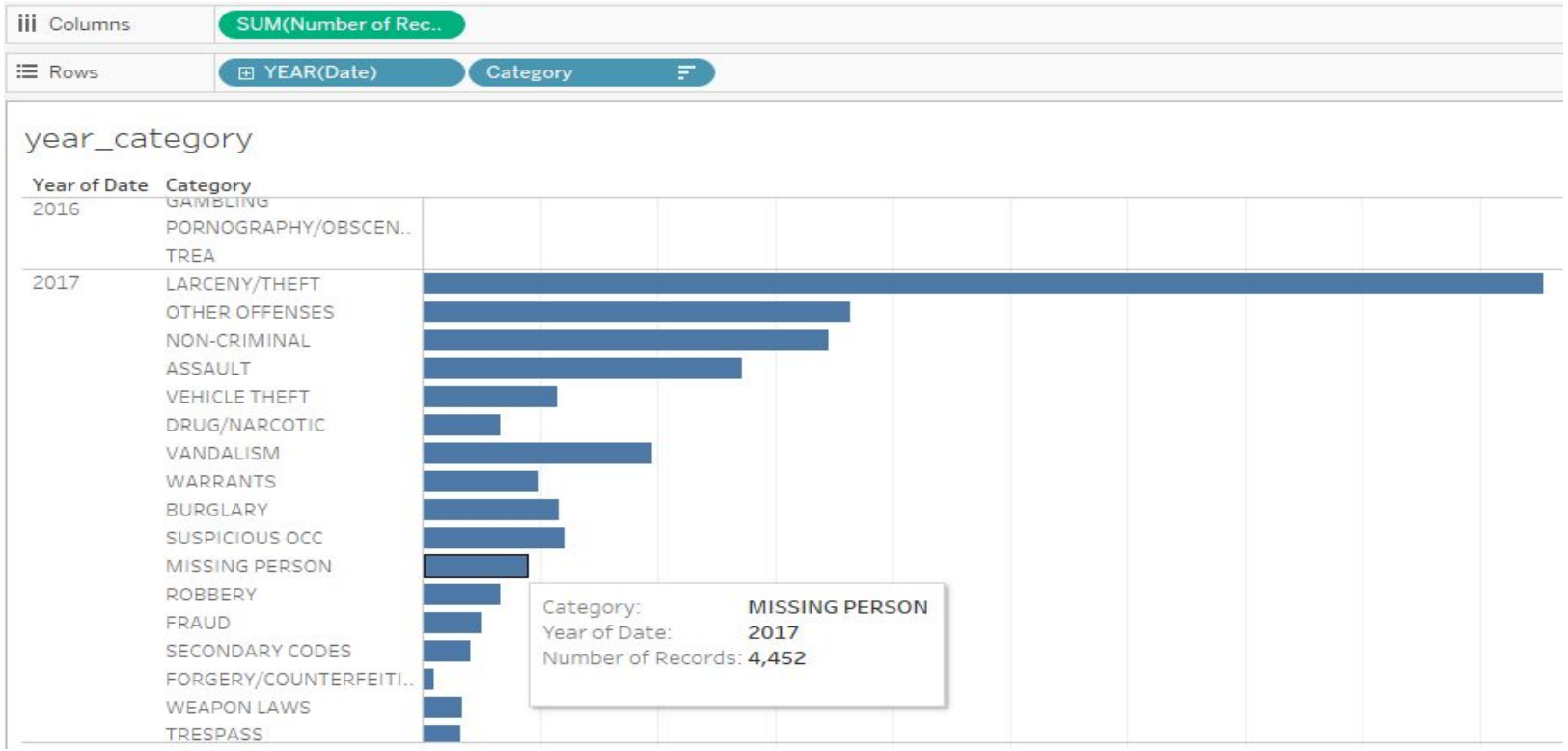

Columns
SUM(Number of Rec..
Rows
YEAR(Date)
Category
year_category
Year of Date
Category
2016
GAMBLING
PORNOGRAPHY/OBSCEN..
TREA
2017
LARCENY/THEFT
OTHER OFFENSES
NON-CRIMINAL
ASSAULT
VEHICLE THEFT
DRUG/NARCOTIC
VANDALISM
WARRANTS
BURGLARY
SUSPICIOUS OCC
MISSING PERSON
ROBBERY
FRAUD
SECONDARY CODES
FORGERY/COUNTERFEITI..
WEAPON LAWS
TRESPASS
Category: MISSING PERSON
Year of Date: 2017
Number of Records: 4,452

Tableau-3

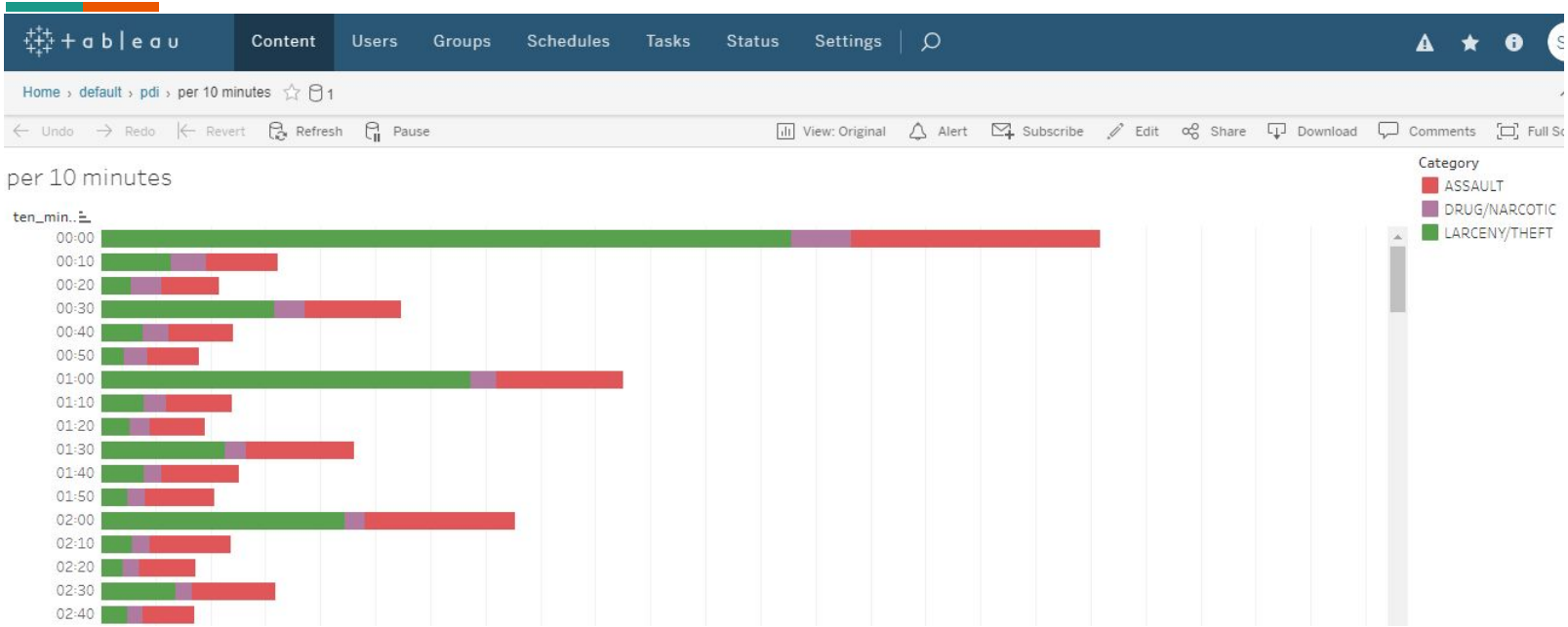
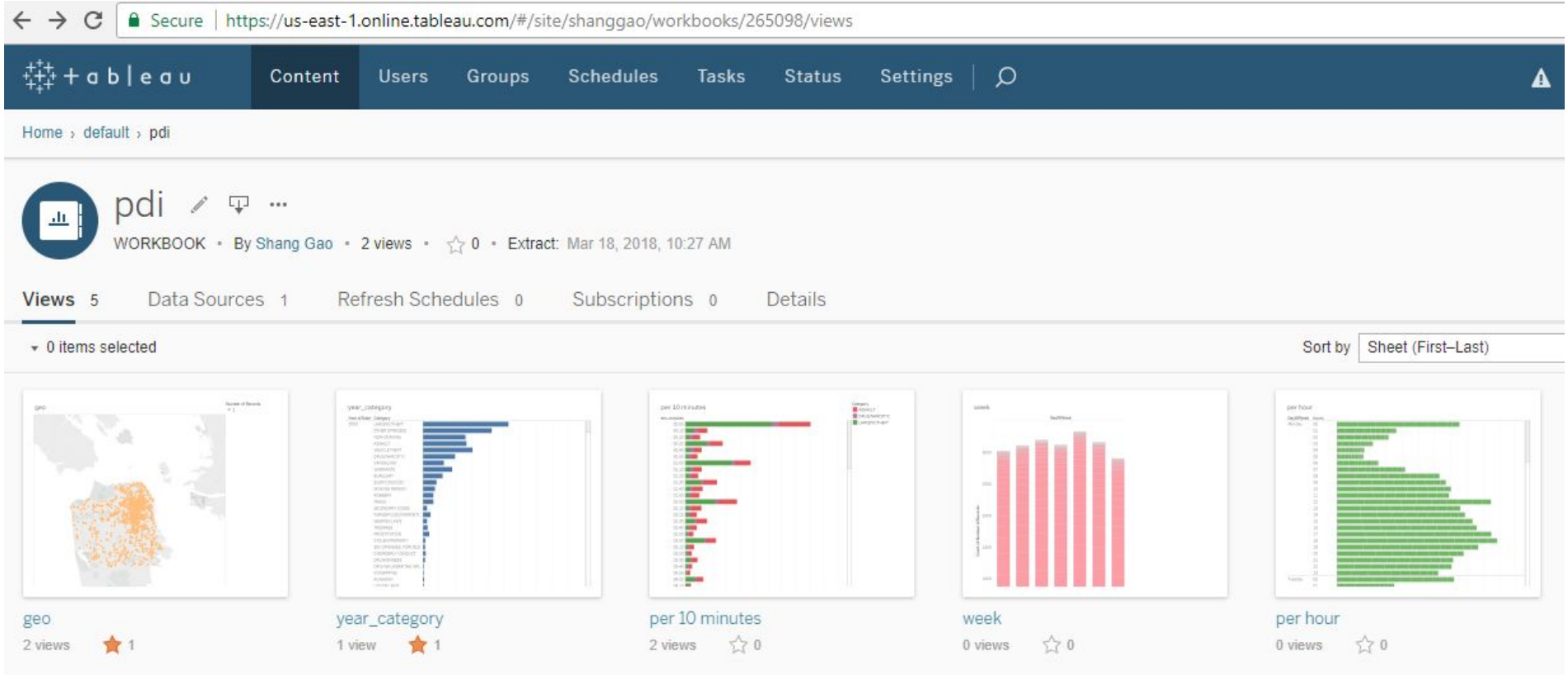

+ableau
Content
Users
Groups
Schedules
Tasks
Status
Settings
Home › default › pdi › per 10 minutes
Undo
Redo
Revert
Refresh
Pause
View: Original
Alert
Subscribe
Edit
Share
Download
Comments
per 10 minutes
ten_min..
00:00
00:10
00:20
00:30
00:40
00:50
01:00
01:10
01:20
01:30
01:40
01:50
02:00
02:10
02:20
02:30
02:40
Category
ASSAULT
DRUG/NARCOTIC
LARCENY/THEFT

Tableau-online



What we learn?

- ➔ How to integrate Spark and Hive
- ➔ How to use tableau to vasulize data
- ➔ Know more about Spark SQL
- ➔ There is some syntax difference between Hql and Spark SQL

Project 3

Text clustering

https://community.cloud.databricks.com/?o=6289929760174924#notebook/2226781146906706/command/2226781146906707

Story

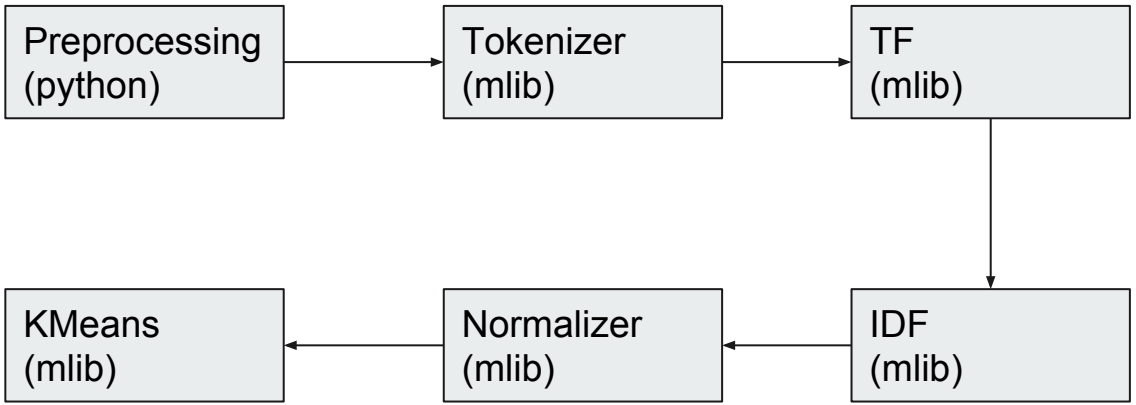
- Clustering corpus into several groups
- 2500 corpus dataset
- https://archive.ics.uci.edu/ml/machine-learning-databases/00217/

Technologies

Kmeans algorithm

1. Partitioning based method
2. Randomly choose k doc as center at initial
3. Compute the distance of each doc to center, and designate each doc to near center.
4. Recompute the the center of each group by means;
5. Iterate step 3&4 until convergence or reach maximum iteration steps(parameter)

Data Pipeline



SparkMlib - scala code

```scala
val sentenceData = spark.sql("select id, text from kmeans_input ").toDF("label", "sentence")

val tokenizer = new Tokenizer().setInputCol("sentence").setOutputCol("words")
val wordsData = tokenizer.transform(sentenceData)
wordsData.show()

val hashingTF = new HashingTF().setInputCol("words").setOutputCol("rawFeatures").setNumFeatures(1000)
val featurizedData = hashingTF.transform(wordsData)
featurizedData.show()

// alternatively, CountVectorizer can also be used to get term frequency vectors
val idf = new IDF().setInputCol("rawFeatures").setOutputCol("idfFeatures")
val idfModel = idf.fit(featurizedData)
val rescaledData = idfModel.transform(featurizedData)
rescaledData.show()

val normalizer = new Normalizer().setInputCol("idfFeatures") .setOutputCol("features").setP(2.0)
val l1NormData = normalizer.transform(rescaledData)
l1NormData.show()

val dataset =l1NormData.select("features")

val kmeans = new KMeans().setK(10).setSeed(1L)
val model = kmeans.fit(dataset)

model.summary.cluster.withColumn("index",functions.monotonically_increasing_id()+1).write.saveAsTable("kmeans_output")
```

SparkMlib - output

1. Within Set Sum of Squared Errors: model.computeCost(dataset)
2. K centers : model.clusterCenters.
3. Give each doc a cluster number: model.summary.cluster

```
1 select label, prediction from kmeans_output
```

▸ (1) Spark Jobs

label	prediction
1	1
2	4
3	4
4	3
5	3
6	3
7	4
8	9
9	9

Command took 4.60 seconds -- by sgao@mum.edu at 3/17/2018, 3:35:57 PM on My Cluster

SparkMlib - Keans Feature

Parameters	Value
k	>1
maxIterations	Default 20
Distance Measure	EuclideanDistanceMeasure
initializationMode	Random or k-means||(default)
initializationSteps	Default 2
epsilon	distance threshold for convergence

For document clustering, the best distance measure should be CosineDistanceMeasure. But this is a new feature in sparkMlib 2.4.0

Pros and Cons of Kmeans

Pros	Cons
Clear model Interpretability	How to choose K?
Suitable use for large dataset	Different initial partitions can result in different final clusters.
Can produce tighter clusters	

Pros and Cons of Spark Mlib

Pros	Cons
Versatility in Feature Extractors, Classification and Regression, Clustering, Association rule	Just have some basic model. Lack variety in models like Density-based Method for clustering, Distance measure etc.
High performance of computation (mahout)	Some deep feature is still not mature enough
Unified api with other spark components	

Conclusion

Before final project we know

Nothing	Something
Spark Streaming	Spark & Spark SQL & Databrick cloud
Spark Mlib	Hive
Apache Kafka	Java
Tableau(Desktop&Online)	Scala
Twitter Stream	Python
Amazon Redshift	

After final project we know

Something	More
Spark Streaming	Spark & Spark SQL & Databrick cloud
Spark Mlib	Hive
Apache Kafka	Java
Tableau(Desktop&Online)	Scala
Twitter Stream	Python
Amazon Redshift	

What we try to do?

- ➔ Spark on Hbase(technical problem)
- ➔ Connect tableau to spark sql(technical problem)
- ➔ Tableau online every minute refresh (license limitation)
- ➔ Kmeans on big dataset(technical problem)
- ➔ Deploy twitter analysis code onto Amazon ec2(time limitation)

References

https://spark.apache.org/

https://spark.apache.org/docs/latest/ml-clustering.html#k-means

https://docs.databricks.com/

https://www.tableau.com/support/help