One University. One World. Yours.

# Master of Science in Computing and Data Analytics

## Managing Information Technology and Systems

## MCDA 5570

**Big Data Group Project**

**Explore the Products Reviews-** ***Final Report***

Under the supervision of

**Nikita Neveditsin**

## Submitted By:


Haozhou Wang (A00431268)

Hemanchal Joshi (A00433394)

Jiye Wang (A00426401)

Naisi Zhen (A00431605)

Rishi Karki (A00432524)

# TABLE OF CONTENTS

# 1. Project Objective

We have got a basic understanding about big data in the last few weeks and learned a lot of usage/APIs of open source software funded by Apache Foundation. But to have a thorough understanding about the big data pipeline, there isn't a better way than getting a hands-on experience by exploring and processing any dataset in the real world.

In this project, we will use a bunch of tools we learned in the class (e.g. HDFS/Hive/Spark/Pig/Zeppelin) to divide the data and explore the real fact behind the numbers.

# 2. Description of Data

Initially we choose data from the Kaggle database.

Link : https://www.kaggle.com/datafiniti/grammar-and-online-product-reviews

It is a list of over 71,045 reviews from 1,000 different products. The dataset includes the text and title of the review, the name and manufacturer of the product, reviewer metadata, and more.

The original dataset targets to analyze the grammar of rating sentences but we also can use it to evaluate the products based on user rating.

## 2.1. Data Exploration

There are many columns in this dataset. Although we may only pick up some of them in our later analysis, we still listed all of their meanings.

The link describes all the available fields in a product record.

https://developer.datafiniti.co/docs/product-data-schema

The description of table schema is illustrated below:

| S.N. | Field Name | Example | Description |
|---|---|---|---|
| 1 | asins | B0009XC FRE | A list of ASINs (Amazon identifiers) used for this product. |
| 2 | brand | Worthingt on | The brand name of this product. |
| 3 | categories | "Home Improvem ent","Heat ers", | A list of category keywords used for this product across multiple sources. |
| 4 | colors | “White”, “Black” | A list of colors available for this product. |
| 5 | count | 30 | The number of units included in the product packaging. Can include a description of the unit. |
| 6 | dateAdded | 2017-01-08T19:12:13Z | The date this product was first added to the product database. |
| 7 | dateupdated | 2017-01-08T19:12:13Z | The most recent date this product was updated or seen by our system. |
| 8 | descriptions | | A list of descriptions from various sources containing:dateSeen,sourceUrls,value |
| 9 | dimension | 23 in x 43.7 in x 30 | The length, width, and height of this product. Units included. |
| 10 | eam | "00140451 25963" | The EAN codes for this product |
| 11 | financingAnd Leasing | | A list of financing or leasing terms associated with this product. |

| | | | |
|---|---|---|---|
| 12 | features | key/value | A list of features associated with this product. |
| 13 | flavours | Berry | A list of flavors available for this product. |
| 14 | imageURLs | https://i5.images.com/asr.jpeg | A list of image URLs for this product. |
| 15 | isbn | 3882215542 | The ISBN code for this product. |
| 16 | keys | "014045125963" | A list of internal Datafiniti identifiers for this product |
| 17 | manufacturer | Worthington | The manufacturer of this product. |
| 18 | manufacture number | 299581 | The manufacturer or model number of this product. |
| 19 | merchants | | A list of merchants selling this product |
| 20 | name | Worthington 20-lb Tank | The product's name. |
| 21 | prices | | A list of prices for this product containing at least 8 attributes |
| 22 | primaryCategories | Electronics | A list of standardized categories to which this product belongs. |
| 23 | primaryImageURLs | | A list of URLs for the primary images taken from each domain sourced in this record. |
| 24 | quantities | | A list of available quantities for this product.Labels like : dateSeen,sourceURLs,value |
| 25 | reviews | | A list of reviews for this product. |

| | | | |
|---|---|---|---|
| 26 | sizes | | A list of sizes available for this product. |
| 27 | skus | | A list of SKUs for this product. SKUs are typically specific to individual retailers or websites. |
| 28 | sourceURLs | | A list of URLs used to generate data for this product. |
| 29 | upc | | The UPC code for this product.. |
| 30 | vin | 1FTMF1E T8EFB00 | The VIN code for this product. |
| 31 | websiteIDs | domain.co m-123 | A list of website IDs for this product. |
| 32 | weight | 20 lbs | The weight of the product. Units included. |

**Table 1:** Description of table schema

## 2.2. Big Data Technologies Used

Following big data technologies were used in our analysis:

### 2.2.1 ETL using PIG

Apache PIG is a high-level language which is used to work on the multiple data process simultaneously. It mainly works with Apache Hadoop and it translates its language to MapReduce job [1]. For this project we used pig as a small portion of Apache Hadoop in order to execute Extract, Transform and Load (ETL) process from a dataset.

### 2.2.2 HIVE for exploring SQL QUERY

Apache HIVE is a component of the Hortonworks Data Platform (HDP) which provides SQL like interface for exploring the dataset stored in the HDP [2] using HDFS. Similarly, it also provides database interfacing to Apache Hadoop.

### 2.2.3 Druid for OLAP Operations

Druid is a distributed datastore which provides high-performance and is well oriented and suited for analytic applications that includes interfacing with the user. Druid is an ideal choice for analytical operations as it performs great for low latency analytics by combining low quantities of a column and provides invert indexing [3].

### 2.2.4 Apache Spark using text-blob for Sentiment Analysis

Apache spark is an analytics engine for the big data which allows the user to achieve high-performance for batch and streaming data [4]. In addition to this we wrote a python script and loaded it into apache spark where the script imported text-blob. Text-blob is a python library used for sentiment analysis and Natural Language Processing.

### 2.2.5 Zeppelin for K-means Clustering

Apache Zeppelin is a multipurpose notebook build under Apache Spark which allows you to explore any datasets and play with the parameters in order to land upon any meaningful findings. So, basically it is an analytical tool but here we use it for unsupervised learning using k-means clustering algorithm.

# 3. Tasks Performed

Following are the tasks performed in our analysis:

## 3.1. Task-1

**Figure out how many records about product AV14LG0R-jtxr-f38QfS ?(PIG)**

Command:

Load Data:

dataSet = LOAD 'hdfs://project/samples.csv' USING PigStorage(',');

productName = FOREACH dataSet GENERATE $0;

result = FILTER productName BY ( $0 == 'AV14LG0R-jtxr-f38QfS');

DUMP result

**The resultant values are two records as shown below**

```
Vertex Stats:
VertexId Parallelism TotalTasks   InputRecords   ReduceInputRecords  OutputRecords  FileBytesRead FileBytesWritten  HdfsByt
esRead HdfsBytesWritten Alias   Feature Outputs
scope-21           1          1           8000                    0              2             0                0       10
972742             52 dataSet,productName,result              hdfs://sandbox-hdp.hortonworks.com:8020/tmp/temp-1548112218
/tmp221828110,

Input(s):
Successfully read 8000 records (10972742 bytes) from: "hdfs:///project/samples.csv"

Output(s):
Successfully stored 2 records (52 bytes) in: "hdfs://sandbox-hdp.hortonworks.com:8020/tmp/temp-1548112218/tmp221828110"

2019-07-30 11:18:14,506 [main] INFO  org.apache.hadoop.mapreduce.lib.input.FileInputFormat - Total input paths to process :
 1
2019-07-30 11:18:14,506 [main] INFO  org.apache.pig.backend.hadoop.executionengine.util.MapRedUtil - Total input paths to p
rocess : 1
```

**Fig 1:** Resultant records

## 3.2. Task-2

**The top 10 products with highest average review rating which have been sold more than 500 times. (Hive)**

1. Use Hive to execute the query:
2. Put the data to the directory:

Hadoop fs -put sampels.csv /projects

3. Import the data into Hive:

There are many ways to create the table in Hive:

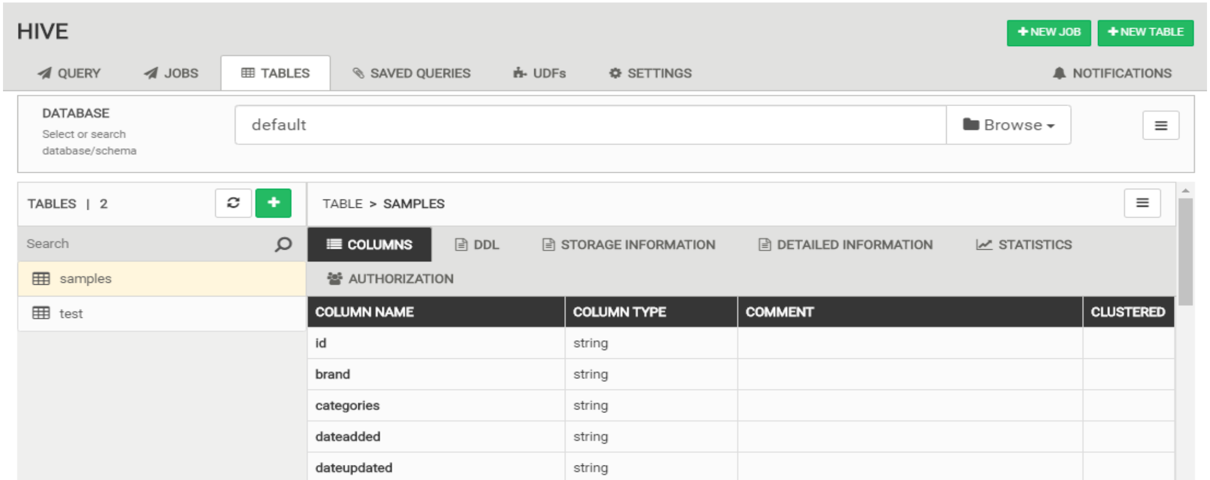
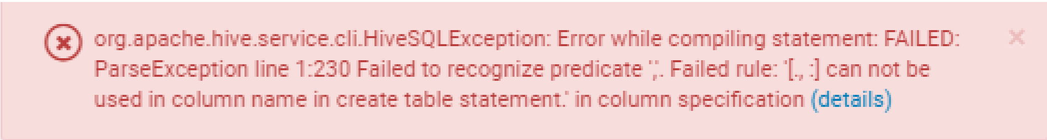



**Fig 2:** Showcasing the table attributes using HIVE

Hive doesn't support column name containing dot, so initially we converted it into underscore (e.g. Change reviews.userCity into reviews_userCity)




Then we observed that the table was created successfully


Then we executed our query: we wanted to know the top 10 products with highest average review rating which has been sold more than 500 times.

Upon execution of SQL Query:

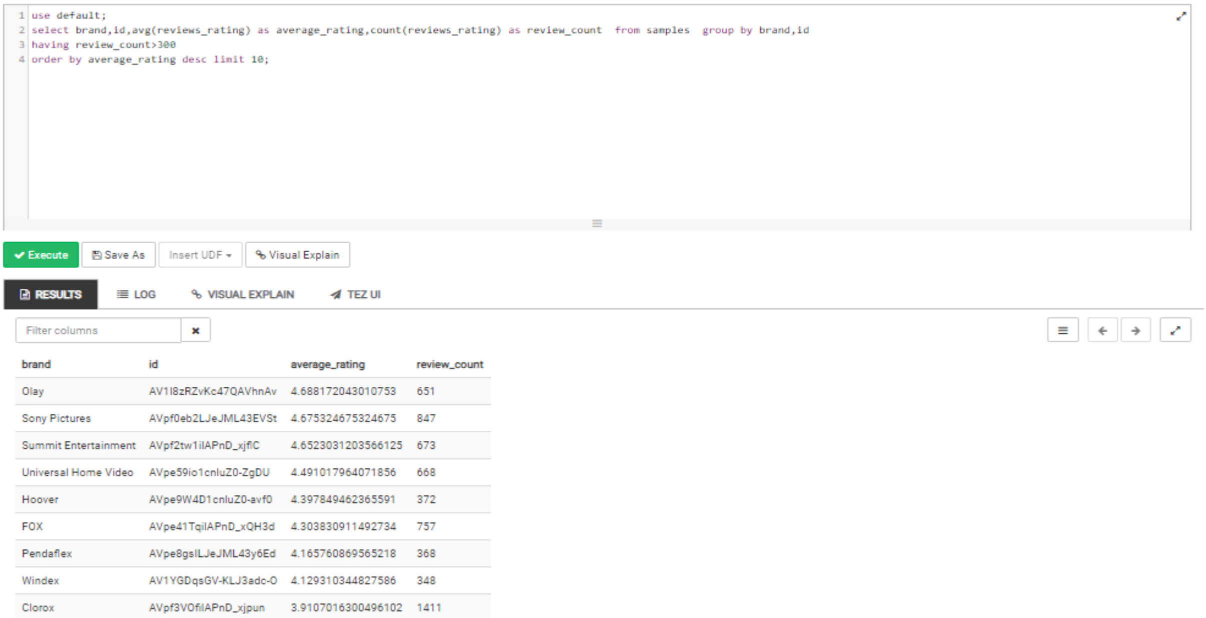
*use default;*

*select brand,id,avg(reviews_rating) as average_rating,count(reviews_rating) as review_count  from samples  group by brand,id*

*having review_count>500*

*order by average_rating desc limit 10;*

We got the following result:




| brand | id | average_rating | review_count |
|---|---|---|---|
| Olay | AV1I8zRZvKc47QAVhnAv | 4.688172043010753 | 651 |
| Sony Pictures | AVpf0eb2LJeJML43EVSt | 4.675324675324675 | 847 |
| Summit Entertainment | AVpf2tw1ilAPnD_xjfIC | 4.6523031203566125 | 673 |
| Universal Home Video | AVpe59io1cnluZ0-ZgDU | 4.491017964071856 | 668 |
| Hoover | AVpe9W4D1cnluZ0-avf0 | 4.397849462365591 | 372 |
| FOX | AVpe41TqilAPnD_xQH3d | 4.303830911492734 | 757 |
| Pendaflex | AVpe8gsILJeJML43y6Ed | 4.165760869565218 | 368 |
| Windex | AV1YGDqsGV-KLJ3adc-O | 4.129310344827586 | 348 |
| Clorox | AVpf3VOfilAPnD_xjpun | 3.9107016300496102 | 1411 |

**Fig 3:** Resultant top 10 products with highest review rating

## 3.3. Task-3

### Carry out OLAP analysis to get records happening in 2015.09.12(Druid)

Apache Druid (incubating) is a real-time analytics database designed for fast slice-and-dice analytics ("OLAP" queries) on large data sets. It could process the data on multiple nodes, so the capacity and speed improved significantly.

1. **Initially we need to follow the same procedure to generate our database.**

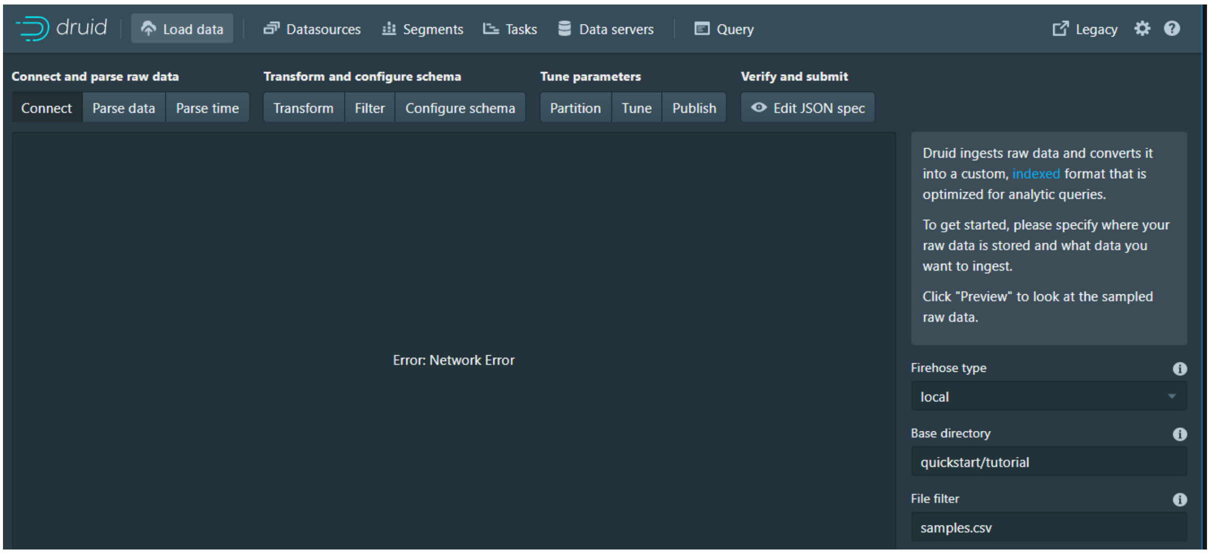



**Fig 4:** Generating the database

**2. From Segments part we can observe that the data is divided into many chunks and imported parallelly**

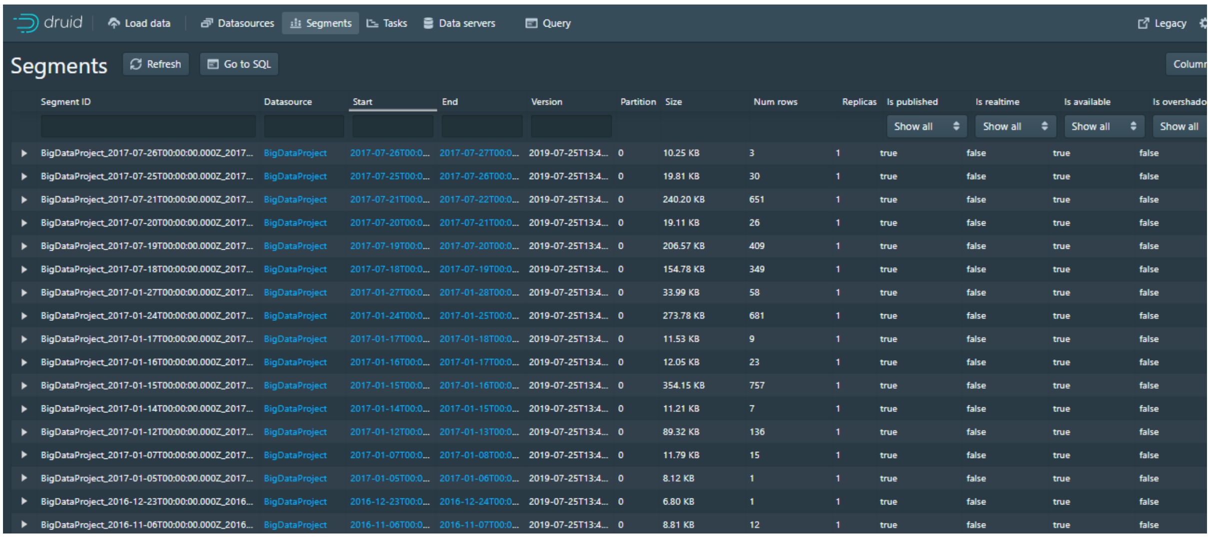



**Fig 5:** Data distribution

**3. Finally if you can get the table created from DataSource table, you can move on to the real analysis part**

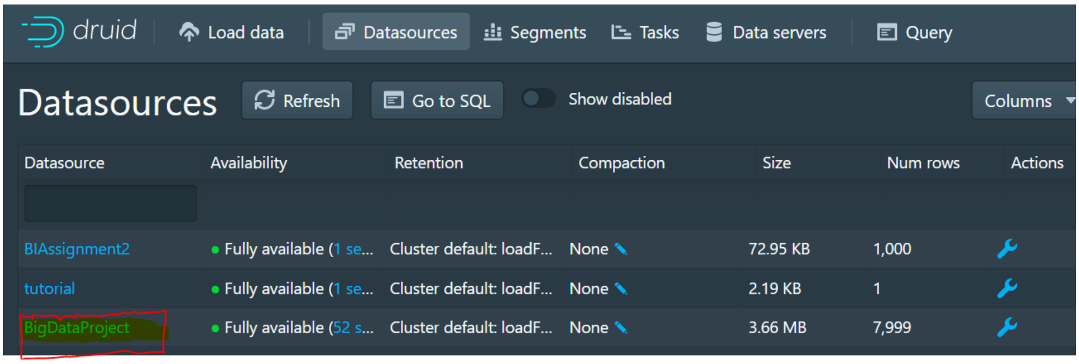



**Fig 6:** Selecting the data source

**4. Execute the analysis:**

There are many operations in OLAP tools like ROLL-UP / DRILL-IN, here we simply wanted to know some metrics (like count) under some dimensions (like the date)

We save the query file into JSON format and save as query.json :

```
{
  "queryType" : "topN",
  "dataSource" : "BigDataProject",
  "intervals" : ["2015-09-12/2015-09-13"],
  "granularity" : "all",
  "dimension" : "name",
  "metric" : "count",
  "threshold" : 10,
  "aggregations" : [
    {
      "type" : "count",
      "name" : "count"
    }
  ]
}
```

And then we executed the query by using command line:

curl -X POST 'localhost:8888/druid/v2/?pretty' -H 'Content-Type:application/json' -H 'Accept:application/json' -d @query.json

We got the result as:

| count | name |
|---|---|
| 34 | Australian Gold Exotic Blend Lotion, SPF 4 |
| 6 | Newman's Own Organics Licorice Twist, Black 5oz |

**Fig 7:** Extracted Result

It means during the period from 2019.05.12-2019.05.13, top 2 selling products are `Australian Gold Exotic Blend Location` and `Newman's Own Organics Licorice Twist` , their counts are 34 and 6 respectively.

## 3.4. Task-4

**User reviews sentiment analysis**

One main usage of NLP is to carry out the sentiment analysis and judge whether the sentence is positive or negative. For this project, we tried to compare the score we got from the review sentence with the review rating.

```
[root@sandbox-hdp ~]# python app.py
19/07/24 14:39:25 WARN NativeCodeLoader: Unable to load native-hadoop library for your platform... using builtin-java classes
 where applicable
Using Spark's default log4j profile: org/apache/spark/log4j-defaults.properties
Setting default log level to "WARN".
To adjust logging level use sc.setLogLevel(newLevel). For SparkR, use setLogLevel(newLevel).
```

**Fig 8:** Loading python script using sandbox

Mainly we used the third-party library(textblob) which is not supported well by zeppelin, so we wrote a native script and submitted it into the job queue.

The core code was scripted in order to know whether the emotion is positive:

*def get_score(sentence):*

*from textblob import TextBlob*

*blob = TextBlob(sentence)*

*return blob.sentences[0].sentiment.polarity*

And also, we used the feature of RDD to execute the script parallelly:

*blist = sentences.map(lambda x: get_score(*

*x[1]) if x[1] is not None else 0).collect()*

In our script, we retrieved the user review score and reviewed sentences by SQL:

*sentences = sqlContext.sql(*

*"""SELECT `reviews.rating`,`reviews.text` FROM table1""").rdd*

After that we got the tendency picture generated by matplotlib:

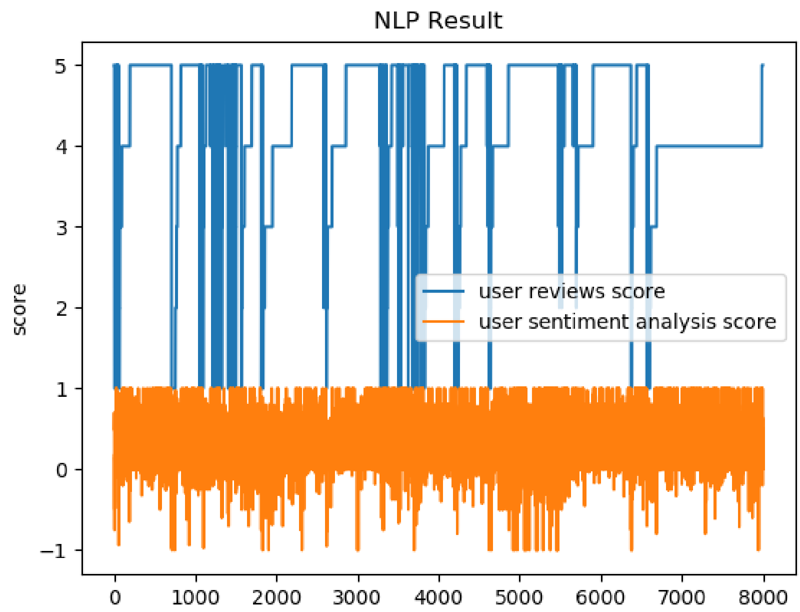



**Fig 9:** The sentiment analysis result

From the picture, we can see that the tendency is clear overall, but in some certain fields it is very confusing. The review score can only have 5 values varying from 1 to 5, while sentiment analysis results are often decimals. If we want to get a more precise result, we would require further knowledge and expertise.

The overall code is available in the Appendix section of this document.

## 3.5. Task-5

**Product K-means Clustering**

The aim of this project is to perform K-means Clustering based the total number of comments, the purchase rate, the recommended rate and average rating.

**Step 1: Data Preparation**

We extract the data using following SQL query:

select count(*) as comments,

sum(case when reviews_didPurchase="TRUE" then 1 else 0 end)/count(*) as purchaseRate,

sum(case when reviews_doRecommend="TRUE" then 1 else 0 end)/count(*) as recommendRate,

AVG(reviews_rating) as rate from productrate group by id;

**Step 2: Perform K-means Clustering with different K values using Zeppelin**

Initially import Machine Learning related packages.

from pyspark.ml.linalg import Vectors

from pyspark.ml.feature import VectorAssembler

from pyspark.ml.clustering import KMeans

from pyspark.ml.evaluation import ClusteringEvaluator

We found that around 7 is the proper K value. The columns c0, c1, c2, c3 represent the number of comments, purchase rate, recommended rate and rating. Following are the first ten results:

```
+---+------+------+------+--------------------+----------+
|_c0|   _c1|   _c2|   _c3|            features|prediction|
+---+------+------+------+--------------------+----------+
|  1|   0.0|   0.0|   5.0|   [1.0,0.0,0.0,5.0]|         0|
|  2|   1.0|   0.0|   5.0|   [2.0,1.0,0.0,5.0]|         0|
|  6|0.8333|0.8333|   5.0|[6.0,0.8333,0.833...|         0|
| 16|  0.25|0.8125|3.8125|[16.0,0.25,0.8125...|         0|
|651|0.0154|0.9324|4.6882|[651.0,0.0154,0.9...|         5|
|348|0.0948|0.7931|4.1293|[348.0,0.0948,0.7...|         3|
|  1|   0.0|   1.0|   5.0|   [1.0,0.0,1.0,5.0]|         0|
| 17|0.6471|0.8235|4.4118|[17.0,0.6471,0.82...|         0|
| 95|0.0526|0.8632|4.2526|[95.0,0.0526,0.86...|         0|
| 25|   0.4|  0.96|  4.76|[25.0,0.4,0.96,4.76]|         0|
+---+------+------+------+--------------------+----------+
```

The silhouette of the results is 0.97608

```
0.976081313506
```

Followings are the clustering centers:

```
[15.3503937   0.18642402  0.63027126  4.32283031]
[8.6060e+03 6.0000e-04 9.9370e-01 4.8214e+00]
[2.283e+03 7.400e-03 3.009e-01 4.456e+00]
[3.70714286e+02 1.77928571e-01 7.39457143e-01 3.97162857e+00]
[1.4235e+03 1.1800e-02 9.5890e-01 4.6944e+00]
[6.00000000e+02 2.79566667e-01 9.03733333e-01 4.46976667e+00]
[1.8860e+03 0.0000e+00 5.9970e-01 4.6039e+00]
```

**Step 3: Visualization**

We generate the scatter plots comparing with different columns.

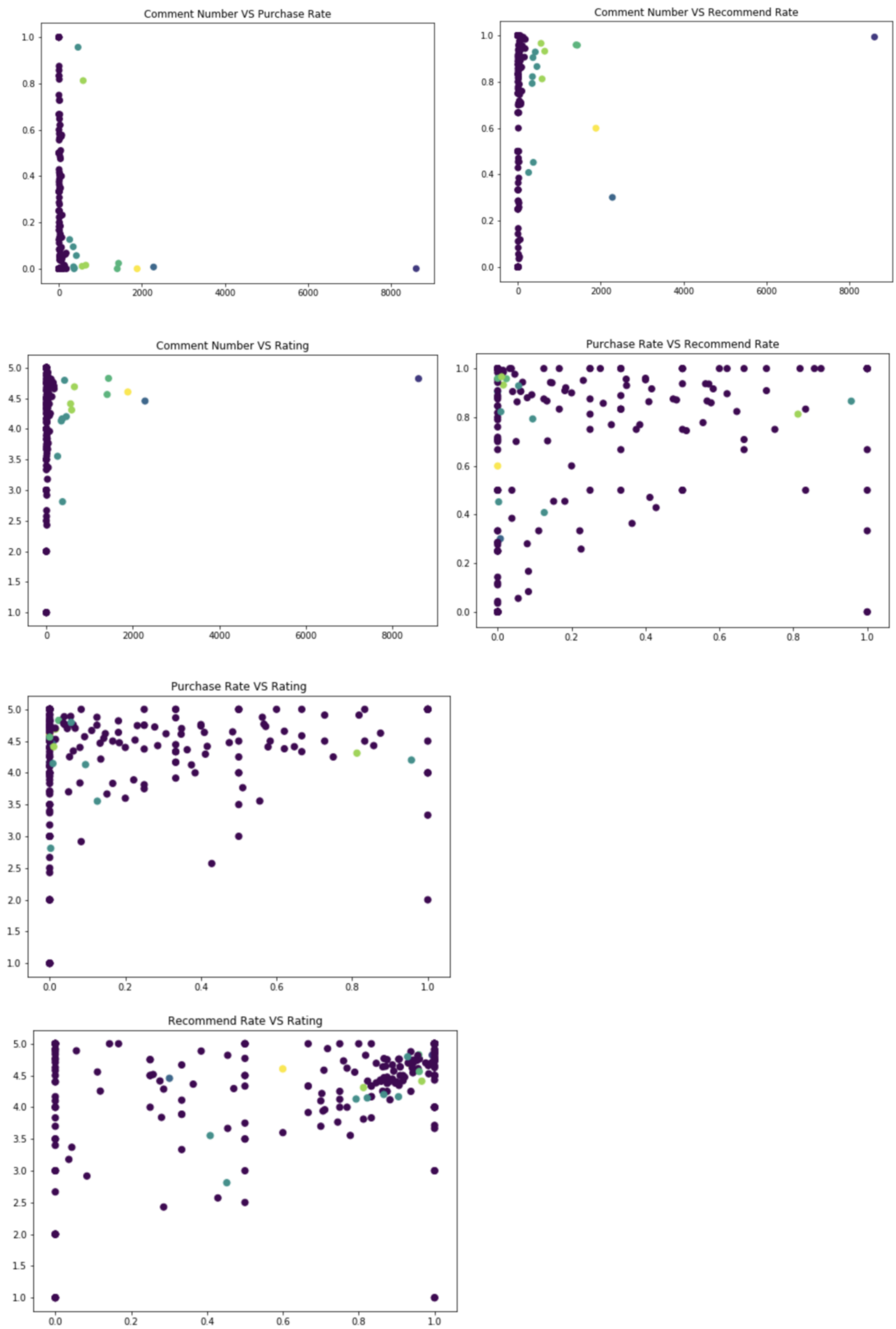



**Fig 10:** Scatter plots representing the clusters

# 4. Findings

- **What was difficult to understand?**

  Hadoop is a huge platform for big data consisting a lot of elements which requires extra time to get familiar with each one of the technologies. The mechanism between the elements make it even harder to get familiar with.

  Significant effort from time to time were spent on data transformation and ingestion tasks.

  Lastly, integrating all Hadoop big data technologies was a challenge and constructive task for the project.

- **What was too easy to understand?**

  Zeppelin interactive notebook style is versatile and easy to use.

  Both Hive and PIG have similar syntax with conventional SQL which is easy to understand and implement.

  Having prior Linux classes made it more helpful and convenient working with several big data technologies using the terminal interface.

- **Maybe something was not covered but you'd like it to be covered?**

  We would rather like to have less topics covered with in depth coverage on any of the latest widely used industry technologies. The other shortcoming from this course what we felt was the live data analysis which is one of the burning topics in Data Science domain.

- **Any other comments are welcome**

  Studying this course and sitting through all the workshops, we found that the difficult part is to understand why big data is efficient, many people talk about it but more real cases will help us better understand it.

  It will be better if there are more theory part (e.g. How Hive works? How does it parse the SQL query into MapReduce job?). API in the documentation can be taught by oneself, but the principle behind the tool is more attractive.

# 5. References

**[1]** Mor, Y. (2019). *Hadoop ETL with Apache Pig*. [online] Xplenty. Available at: https://www.xplenty.com/blog/etl-on-hadoop-with-apache-pig/ [Accessed 25 Jul. 2019].

**[2]** Hortonworks. (2019). *How to Process Data with Apache Hive - Hortonworks*. [online] Available at: https://hortonworks.com/tutorial/how-to-process-data-with-apache-hive/ [Accessed 25 Jul. 2019].

**[3]** Hortonworks. (2019). Ultra-fast OLAP Analytics with Apache Hive and Druid - Part 1 of 3 - Hortonworks. [online] Available at: https://hortonworks.com/blog/apache-hive-druid-part-1-3/ [Accessed 26 Jul. 2019].

**[4]** Spark.apache.org. (2019). *Apache Spark™ - Unified Analytics Engine for Big Data*. [online] Available at: https://spark.apache.org/ [Accessed 26 Jul. 2019].

# 6. Appendix A [Script for Task 1]

***import matplotlib.pyplot as plt***

***import matplotlib***

***import numpy as np***

"""

This script is used to analyze the relationship between user review score and sentiment score

Prerequisite:

put the **samples.csv** file into hdfs directory: hdfs:///project/samples.csv

Install third-party dependencies:

>>> ***pip install pyspark***

>>> ***pip install textblob***

download dataset:

>>> ***python -m textblob.download_corpora***

Submit the script into spark job queue:

>>> ***python app.py***

*"""*

***from pyspark.sql import SparkSession***

***from pyspark import SQLContext, SparkConf***

***def get_score(sentence):***

***""" carry out the sentiment analysis***

***"""***

***from textblob import TextBlob***

***blob = TextBlob(sentence)***

```
    return blob.sentences[0].sentiment.polarity
spark = SparkSession.builder.master(
    "local").appName("Word Count").getOrCreate()
df = spark.read.format("csv").option("header", "true").load("hdfs:///project/samples.csv")
sqlContext = SQLContext(spark)
sqlContext.registerDataFrameAsTable(df, "table1")
sentences = sqlContext.sql(
    """SELECT `reviews.rating`,`reviews.text` FROM table1""").rdd
sentences.collect()
alist = sentences.map(lambda x: float(
    x[0]) if x[0] is not None else 0).collect()
blist = sentences.map(lambda x: get_score(
    x[1]) if x[1] is not None else 0).collect()
plt.switch_backend('agg')
plt.plot(alist, label="user reviews score")
plt.plot(blist, label="user sentiment analysis score")
plt.title('NLP Result')
plt.ylabel('score')
plt.legend()
plt.savefig("temp.png")
with open("x1.txt", "w+") as input_:
    input_.write(','.join([str(x) for x in blist]))
with open("review.txt", "w+") as input_:
    input_.write(','.join([str(x) for x in alist]))
```

# 7. Appendix B [Install Apache Druidr]

1. Download the binary package:

***Wget https://www-eu.apache.org/dist/incubator/druid/0.15.0-incubating/apache-druid-0.15.0-incubating-bin.tar.gz***

2. Extract Druid by running the following commands in your terminal:

***tar -xzf apache-druid-0.15.0-incubating-bin.tar.gz***

***cd apache-druid-0.15.0-incubating***

3. Download zookeeper

***curl https://archive.apache.org/dist/zookeeper/zookeeper-3.4.11/zookeeper-3.4.11.tar.gz -o zookeeper-3.4.11.tar.gz***

***tar -xzf zookeeper-3.4.11.tar.gz***

***mv zookeeper-3.4.11 zk***

4. Start the Apache Druid Server

***./bin/start-micro-quickstart***

5. Open the browser address [http://localhost:8888](http://localhost:8888) and view the application

# 8. Appendix C [Script for Task 5]

```
%spark2.pyspark
from pyspark.ml.linalg import Vectors
from pyspark.ml.feature import VectorAssembler
from pyspark.ml.clustering import KMeans
from pyspark.ml.evaluation import ClusteringEvaluator
from pyspark.ml import Pipeline
import matplotlib.pyplot as plt
from numpy import genfromtxt
df = spark.read.format("csv").option("inferSchema",
"true").load("/__dsets/bigdata/productRateN.csv")


va = VectorAssembler(
   inputCols=["_c0","_c1","_c2", "_c3"],
   outputCol="features")


kmeans = KMeans().setK(5).setSeed(1L)


pipeline = Pipeline(stages=[va, kmeans])


model = pipeline.fit(df)


res = model.transform(df)
res.show(5)
```

```
pred = res.toPandas()



# Evaluate clustering by computing Silhouette score
evaluator = ClusteringEvaluator()
silhouette = evaluator.evaluate(res)
print(silhouette)

# Shows the result.
centers = model.stages[-1].clusterCenters()
print("Cluster Centers: ")
for center in centers:
    print(center)

fig1 = plt.figure()
ax1 = fig1.add_subplot(111)
ax1.set_title("Comment Number VS Purchase Rate")
plt.scatter(pred["_c0"], pred["_c1"], s = 50, c = pred["prediction"])
plt.show()
fig2 = plt.figure()
ax2 = fig2.add_subplot(111)
ax2.set_title("Comment Number VS Recommend Rate")
plt.scatter(pred["_c0"], pred["_c2"], s = 50, c = pred["prediction"])
plt.show()
```

```
fig3 = plt.figure()

ax3 = fig3.add_subplot(111)

ax3.set_title("Comment Number VS Rating")

plt.scatter(pred["_c0"], pred["_c3"], s = 50, c = pred["prediction"])

plt.show()


fig4 = plt.figure()

ax4 = fig4.add_subplot(111)

ax4.set_title("Purchase Rate VS Recommend Rate")

plt.scatter(pred["_c1"], pred["_c2"], s = 50, c = pred["prediction"])

plt.show()


fig5 = plt.figure()

ax5 = fig5.add_subplot(111)

ax5.set_title("Purchase Rate VS Rating")

plt.scatter(pred["_c1"], pred["_c3"], s = 50, c = pred["prediction"])

plt.show()


fig6 = plt.figure()

ax6 = fig6.add_subplot(111)

ax6.set_title("Recommend Rate VS Rating")

plt.scatter(pred["_c2"], pred["_c3"], s = 50, c = pred["prediction"])

plt.show()
```