Amazon Review Analytics

I. **Group Name**: **SPARTAN**

II. **Team Members**:

Rima Halder,

Rajesh Kusuma,

Pavan Sai Reddy Koduru.

Guided by: **Dr. Jun Wang**.

***Problem*— Julian McAuley's lab at Stanford University, as part of their research, collected 80 million Amazon reviews from 1995 to 2013. This mammoth dataset is divided into different categories.**

For example, there is a manager at a media company and want to start a new product line to boost the revenue of the company. Being a media company, they want to get into either eBooks, movies, TV shows, or similar such product categories in the entertainment industry.

Since starting a new product line, they want to be sure about the choice of products that they buy (and sell). You have three options of product categories to choose from - CDs and Vinyl, Movies and eBooks (Kindle).

III. Objective

The company manager uses the gigantic Amazon dataset and focus on few key metrics to identify which product category is correct for investing. The manager of the company focuses on few datapoints as below.

- Which product category has a larger market size?
- Which product category is likely to make the customer happy after the purchase?
- Which product category is likely to be purchased heavily based on grant total of helpful ratio?
- Which product category is likely to be purchased heavily based on average help ratio?
- Which product category is famous across reviewers based on length of the review text?
- Which product category is famous across high rated reviewers based on length of the review text?
- Which product category is famous based on positive key words in summary text?

They will choose the best product category among all the three that satisfy most of the key points.

IV. Dataset Used

- Amazon Data: http://jmcauley.ucsd.edu/data/amazon/

V. Sample Data

A. Data Format

Column Name	Value
reviewerID	A2SUAM1J3GNN3B
asin	0000013714
reviewerName	J. McDonald
helpful	**": [** 2, 3 **]"**
reviewText	"I bought this for my husband who plays the piano. "
overall	5
summary	Heavenly Highway Hymns
unixReviewTime	1252800000
reviewTime	09 13, 2009

Fig 1. Data Format

B. Description

- **`reviewerID`** - ID of the reviewer, e.g. A2SUAM1J3GNN3B
- **`asin`** - ID of the product, e.g. 0000013714
- **`reviewerName`** - name of the reviewer
- **`helpful`** - helpfulness rating of the review, e.g. 2/3
- **`reviewText`** - text of the review
- **`overall`** - rating of the product
- **`summary`** - summary of the review
- **`unixReviewTime`** - time of the review (unix time)
- **`reviewTime`** - time of the review (raw)

We focus on three columns for the entire analysis.

- **overall**
- **reviewText**
- **helpful.**

VI. SOLUTION

We use multiple AWS (Amazon Web Service) technologies for the entire project execution starting from configuring the set up and platform to the analysis of the data which leads to the result.

A. *AWS Educate Subscription*

AWS Educate is Amazon's global initiative to provide students comprehensive resources for building skills in the cloud. It is a no-cost curriculum providing access to content, training, pathways, AWS services, and the AWS Educate Job Board with employment opportunities.

We have taken AWS educate student subscription to use the Amazon tools for this project.

https://aws.amazon.com/education/awseducate/

There is no cost to join and AWS Educate provides hands-on access to AWS technology, training resources, course content and collaboration forums. Students and educators apply online at www.awseducate.com to access: Grants for free usage of AWS services. Labs, tutorials, and training on AWS products.

B. *Amazon S3 (Simple Storage Solution)*

- An Amazon S3 bucket is a public cloud storage resource available in Amazon Web Services' (AWS) Simple Storage Service (S3), an object storage offering. Amazon S3 buckets, which are similar to file folders, store objects, which consist of data and its descriptive metadata.
- To Access Amazon S3, you need to follow below steps.
- Sign in to the AWS Management Console and open the S3 console at https://console.aws.amazon.com/s3/
- In the Buckets list, choose the name of the bucket that you want to upload your files to.
- Create a folder named Amazon-Review
- Choose Upload.
- On the Upload page, choose Add files or Add folder.

So, we upload three datasets for (Movies & Tv, CDs & Vinyl & Kindle Store) which we would analyze for this project.

- reviews_CDs_and_Vinyl_5.json.gz
- reviews_Kindle_Store_5.json.gz
- reviews_Movies_and_TV_5.json.gz

Finally, we upload the rstudio emr configuration to launch R Studio in Server and in chrome for the final analysis. We can download the information from AWS Blogs.

- rstudio_sparklyr_emr5.sh

Example of Configuration Commands inside .sh file.as in (1a,1b)

```
sudo rstudio-server stop || true        (1.a)
sudo rstudio-server                     (1.b))
```

C. *Amazon EMR (Elastic Map Reduce)*

Amazon EMR is the industry-leading cloud big data platform for processing vast amounts of data using open source tools such as Apache Spark, Apache Hive, Apache HBase, Apache Flink, Apache Hudi, and Presto.

It uses Hadoop, an open source framework, to distribute your data and processing across a resizable cluster of Amazons EC2 instances.

Amazon EMR processes big data across a Hadoop cluster of virtual servers on Amazon Elastic Compute Cloud (EC2) and Amazon Simple Storage Service (S3). The elastic in EMR's name refers to its dynamic resizing ability, which allows it to ramp up or reduce resource use depending on the demand at any given time.

Below are few steps for starting EMR cluster for processing big data.

- Sign in to the AWS Management Console and open the Amazon EMR console at https://console.aws.amazon.com/elasticmapreduce/
- Choose Create cluster.
- On the Create Cluster -Go to Advance options,
- Select emr 5.29
- Add Spark as extra in the Software Configuration.
- Enter a Cluster name that helps you identify the cluster, for example, My First EMR Cluster.
- Select spark s3://adb-amazon-review/Amazon-Review/rstudio_sparklyr_emr5.sh in bootstrap action and save.
- Select Create cluster
- The cluster will get started in few minutes.

D. Add Security Group

- A security group acts as a virtual firewall for your EC2 instances to control incoming and outgoing traffic.
- Inbound rules control the incoming traffic to your instance.
- Outbound rules control the outgoing traffic from your instance

- EMR will create two security groups for the cluster, one for **the master node (which hosts the Thrift service)** and one for the **slave nodes**
- Security groups are stateful — if you send a request from your instance, the response traffic for that request is allowed to flow in regardless of inbound security group rules.

We need to add Inbound rules to the "**Security Groups for Master**" after we create the EMR cluster.

The steps are as below

- Open the Amazon EMR console at https://console.aws.amazon.com/elasticmapreduce/
- Choose Clusters.
- Choose the Name of the cluster.
- Under Security and access choose the Security groups for Master link..
- Choose ElasticMapReduce-master from the list.
- Scroll to the bottom of the list of rules and choose Add Rule.
- For Type, select Custom Type.
- For Port, add 8787
- For source, select My IP.
- This automatically adds the IP address of your client computer as the source address.
- Choose Save.

This inbound rule allows to launch Rstudio in chrome along with Public DNS address copied from EC2 instance.

Find the table below **(Fig 2)** to refer to add inbound rule with port 8787 as mentioned above.

Type	Protocol	Port	Source
Custom TCP	TCP	8787	My IP

Fig. 2. Example of an Inbound Rule.

E. Amazon EC2

- An EC2 instance is nothing but a virtual server in Amazon Web services terminology.
- It stands for Elastic Compute Cloud. It is a web service where an AWS subscriber can request and provision a compute server in AWS cloud.
- EMR is a collection of EC2 instances with Hadoop (and optionally Hive and/or Pig) installed and configured on them.

Below are the few steps that helps to collect the public DNS address from EC2 instances.

- Open the Amazon Ec2 console at https://console.aws.amazon.com/ec2/
- In the navigation pane, choose Instances.
- Select your Running instance from the list.
- In the details pane, the Public DNS (IPv4) and
- Private DNS fields display the DNS hostnames.

F. SparkR (R on Spark)

SparkR is an R package that provides a light-weight frontend to use Apache Spark from R. In Spark 3.0.1, SparkR provides a distributed data frame implementation that supports operations like selection, filtering, aggregation etc. (similar to R data frames, dplyr) but on large datasets. SparkR also supports distributed machine learning using MLlib.

1) SparkDataFrame

A SparkDataFrame is a distributed collection of data organized into named columns. It is conceptually equivalent to a table in a relational database or a data frame in R, but with richer optimizations under the hood. SparkDataFrames can be constructed from a wide array of sources such as: structured data files, tables in Hive, external databases, or existing local R data frames.

2) SparkSession

- The entry point into SparkR is the SparkSession which connects your R program to a Spark cluster.
- You can create a SparkSession using sparkR.session and pass in options such as the application name, any spark packages depended on, etc.
- Further, you can also work with SparkDataFrames via SparkSession.
- If you are working from the sparkR shell, the SparkSession should already be created for you, and you would not need to call sparkR.session.

3) Starting SparkR from Rstudio

- We launch the Rstudio in chrome.
- Launch the Terminal
- Launch a SparkR shell in Terminal
- Get the SparkContext
- Load the datasets from S3 path.
- Start the final analysis.

Next, we start with our coding and final data analysis part. Till now we create the entire environment and get the necessary tools for the project execution.
Loading the spark Context and spark session as in (2.a,2.b)

```
library(SparkR)                          2.a
sc = sparkR.session(master='local')      2 b
```

AWS makes the environment ready in few minutes. we can instantly build up large Hadoop clusters which will start processing within minutes, not hours or days.

VII. ANALYSIS

- **Data analysis is executed based on the key metrics.**
- **All the calculations are shown based on any one of the datasets.**
- **Same analysis and calculations applied to three datasets.**

*1) **Which product category has a larger market size ?***

a) Define the S3 path in a variable

S3_BUCKET_NAME<- "s3://adb-amazon-review/Amazon-Review/"

b) Load all the three datasets from S3. Each of the below query loads the dataset from the S3 bucket as a Spark DataFrame.

ar_Movies_and_TV<-
SparkR::read.df(path=paste(S3_BUCKET_NAME,"/reviews_Movies_and_TV_5.json", sep = ""), source="json")

ar_CDs_and_Vinyl<-
SparkR::read.df(path=paste(S3_BUCKET_NAME,"/reviews_CDs_and_Vinyl_5.json", sep = ""), source="json")

ar_Kindle_Store<-
SparkR::read.df(path=paste(S3_BUCKET_NAME,"/reviews_Kindle_Store_5.json", sep = ""), source="json")

c) Find number of Rows & Number of Columns for each dataset using following queries as **Fig 3.** dim in SparkR returns the dimensions (number of rows and columns) of a SparkDataFrame.

d) Sum up total number of reviews from all three products using queries as **Fig 4.**

e) Calculate the percentage of total review for each product type as **Fig 4.**

Input Query	# Rows	#Columns
dim_cd<- dim(ar_CDs_and_Vinyl)	1097592	9
dim_cd<- dim(ar_CDs_and_Vinyl)	982619	9
dim_movies<- dim(ar_Movies_and_TV)	1697533	9

Fig. 3 Find number of Rows & Number of Columns

Query	Output
Total_reviews <- dim_cd[1] + dim_kindle[1] + dim_movies[1]	3777744
msize_Movies_and_TV<- (dim_movies[1]/Total_reviews)*100	**44.9351%**
msize_CDs_and_Vinyl<- (dim_cd[1]/Total_reviews)*100	29.05417%
msize_Kindle_Store<- (dim_kindle[1]/Total_reviews)*100	26.01074%

Fig. 4 Find Total and Individual review %

Result: Movies and TV product category has largest market size i.e. 45% among three.

*2) **Which product category is likely to make the customer happy after the purchase?***

a) Our Assumption is, customers are happy if the rating is equal or greater than four i.e. =>4?

b) First Group each dataset by the column field "overall". "Overall" is a column in the review data.

Example of Cds & Vinyl (Group by)

```
count_overall_rating_CDs_and_Vinyl<-
summarize(groupBy(ar_CDs_and_Vinyl,
ar_CDs_and_Vinyl$overall),
                                count=
count(ar_CDs_and_Vinyl$overall))
```

c) Finally Filter the output of group By and find the total count of review rows having rating >=4.

Queries:

Example of Cds & Vinyl (Filter)

```
high_rating_CDs_and_Vinyl<-
filter(count_overall_rating_CDs_and_Vinyl,

count_overall_rating_CDs_and_Vinyl$overall>=4
)
```

Output is shown in Fig 5 & Fig 6.

"Overall" column	Count
1.0	46195
2.0	246326
3.0	101824
4.0	46571
5.0	656676

Fig 5 (Output of Group by "overall")

"Overall" column	Count
4.0	46571
5.0	656676

Fig 6. (Output of Filter)

d) Find and aggregate/sum count of rows for each of the product having "overall">=4. We need to sum the filtered count in the previous result.

Example of a query for CDs & Vinyl:

showDF(agg(high_rating_Movies_and_TV,tot= sum(high_rating_Movies_and_TV$count))

- **agg** : Aggregates and sum up on the entire DataFrame without groups.

- **showDF**: It display a spark data frame in a table format.

Product Type	Output
CDs & Vinyl	903002
Kindle Store	829277
Movies & TV	1289602

Fig 7: Total Count of overall >=4

Result: The product category is likely to make customer happy is Movies and TV segment.

*3) **Which product category is likely to be purchased heavily based on grant total of helpful ratio?***

Helpful ratio = number of people who found the review helpful/ total number of people

a) Extract 1st and 2n values from helpful column to find helpful ratio using posexplode & filter for each dataset.

Example Query for Movies & TV

temp_Movies_and_TV<- select(ar_Movies_and_TV,posexplode(ar_Movies_and_TV $helpful))

temp_Movies_and_TV_helpful<- filter(temp_Movies_and_TV,temp_Movies_and_TV$pos == 0)
temp_Movies_and_TV_totalreview<- filter(temp_Movies_and_TV,temp_Movies_and_TV$pos == 1)

- **posexplode** : Creates a new row for each element with position in the given array or map column.

b) Collects all the elements of a SparkDataFrame and coerces them into an R data.frame.

Example Query of Movies & Tv

temp_Movies_and_TV_helpful_r_df<- collect(temp_Movies_and_TV_helpful)

temp_Movies_and_TV_totalreview_r_df<- collect(temp_Movies_and_TV_totalreview)

- **collect**: Collects all the elements of a SparkDataFrame and coerces them into an R data.frame.

c) Add helpful ratio and total Reviews in each row in the previously generated R.data.frame as "tot" & "fnd_helpful" fields and also calculate & add "helpful_ratio" as another field.

Example Query for Movies & TV

temp_Movies_and_TV_helpful_r_df$tot<- temp_Movies_and_TV_totalreview_r_df$col

colnames(temp_Movies_and_TV_helpful_r_df)[2]<- "fnd_helpful"

temp_Movies_and_TV_helpful_r_df$helpful_ratio<- temp_Movies_and_TV_helpful_r_df$fnd_helpful/ temp_Movies_and_TV_helpful_r_df$tot

d) Find grant total of helpful ratio by summing up helpful_ratio in each row

Example Query for Movies & TV

Movies_and_TV_helpful_r_total_help_ratio<- dplyr::summarize(temp_Movies_and_TV_helpful_r_df,sum (temp_Movies_and_TV_helpful_r_df$helpful_ratio,na.rm = TRUE))

- **dplyr :** We install this package in Rstudio Packages. dplyr is a package for making data manipulation easier.

- **summarise():** It creates a new data frame. It will have one (or more) rows for each combination of grouping variables; if there are no grouping variables, the output will have a single row summarising all observations in the input. It will contain one column for each grouping variable and one column for each of the summary statistics that we have specified.

Output:

Dataset	Helpful Ratio
Movies_and_TV_helpful_r_total_help_ratio	**678481.9**
CDs_and_Vinyl_helpful_r_total_help_ratio	566979.2
Kindle_Store_helpful_r_total_help_ratio	367361.5

Fig 8: grant total of Helpful ratio

Result: Movies and TV has got higher helpful ratio i.e.678481.9, hence Movies and TV is likely to be purchased heavily

*4) **Which product category is likely to be purchased heavily based on average help ratio?***

a) This analysis is same as before. Just like the previous calculation, we calculate the mean/average instead of summing up the helpful ratios in each row.

Example Query of Kindle Store:

```
Kindle_Store_helpful_r_avg_help_ratio<-
dplyr::summarize(temp_Kindle_Store_helpful_r_df,
mean(temp_Kindle_Store_helpful_r_df$helpful_ratio,na.rm
= TRUE))
```

Output:

Helpful ratio variable	Output
Movies_and_TV_helpful_r_total_help_ratio	0.6231739
CDs_and_Vinyl_helpful_r_total_help_ratio	0.7073386
Kindle_Store_helpful_r_total_help_ratio	0.8122255

Fig 9: Average of Helpful ratio

Result: kindle store is famous product line based on average helpful ratio

*5) **Which product category is famous across reviewers based on length of the review text?***

- Our Assumption is, If length of the review text is more, then customers are happy.
- Add length of the review text as a new column to the for each review in each product category data frame
- Aggregate the review text length column with sum function to get the total length of review text for each product category.

a) Calculate the length of each reviewText for each row and add a new field or column as "reviewtext_length" in the original dataset for each category.

Example query for Movies & Tv:

```
ar_Movies_and_TV$reviewtext_length<-
length(ar_Movies_and_TV$reviewText)
```

b) Sum up the reviewText_length for all the rows in Movies dataset and return the output.

Example query for Movies & Tv:

```
head(agg(ar_Movies_and_TV,length_tol=
sum(ar_Movies_and_TV$reviewtext_length)))
```

Dataset	Output
Movies & TV	1565297599
CDs & Vinyl	1089459822
Kindle Store	593450774

Fig 10: Total Review Text length for each dataset

Result: As per the output of review length, Movies and TV is the product is famous across reviewers.

*6) **Which product category is famous across high rated reviewers based on length of the review text?***

- Assumption: If length of the review text is more & rating is 5, then customers are happy.

a) Filter the data with length of the review text & reviews got 5 ratings for each of the datsets

Example Query for Movies & TV:

```
high_rating_movies_TV_rev_txt_tol_len<-
filter(ar_Movies_and_TV, ar_Movies_and_TV$overall==5
)
```

b) Sum up the reviewText_length for all the filtered rows rows in previous output.

Example Query for Movies & TV:

showDF(agg(high_rating_movies_TV_rev_txt_tol_len, tot = sum(high_rating_movies_TV_rev_txt_tol_len$reviewtext_length)))

Dataset	Output
Movies & TV	712575303
CDs & Vinyl	622567752
Kindle Store	319591343

Fig: 11 Total review text length with rating=5

Result : Movies and TV is the product is famous across top rated reviews.

*7) **Which product category is famous based on positive key words in summary text ?***

Assumption: If summary text contains below keywords, then it is assumed customers are happy

Positive Keywords:

- **Good**
- **Nice**
- **Excellent**
- **Happy**
- **Satisfied**
- **Worth**

a) These queries use RDD API to filter the data using positive keywords (good/nice/excellent/happy/satisfied/worth) using "grepl"

b) Convert RDD to DataFrame and count positive reviews received for each of the product category.

Example query for Movies & Tv.

```
ar_Movies_and_TV.rdd.filtered <-
SparkR:::filterRDD(ar_Movies_and_TV.rdd, function(s)
{ grepl("good|nice|excellent|happy|satisfied|worth",
s$summary, ignore.case = TRUE, perl = TRUE) })
```

grepl:
grep, grepl, search for matches to argument pattern within each element of a character vector: they differ in the format of and amount of detail in the results.

Below table in **(Fig 10)** differentiate RDD vs Dataframe for analyzing data.

RDD	Dataframe
RDD stands for resilient distributed datasets. It is the basic data structure defined by Spark so the data can be distributed among cluster nodes.	Comparing to RDD, Dataframe introduces the concept of schema, which refers to the structure of data.
RDD contains large information, we can apply actions to RDD to return values, and transformations to return new RDD.	Dataframe allows Spark to manage the structure and only send the data between nodes, which is more efficient than Jave serialization.
It is simply a set of Java or Scala objects representing data. Spark distributes data within a cluster using Java serialization, which is a process of converting an object to a series of bytes. Commonly used RDD transformation include map, filter, reduce, and reduceByKey	Dataframe API is a lot like relational queries. We can use various relational operations that are similar to SQL expression.

Fig 12. RDD vs Dataframe in Spark

Output:

Dataset	Output
Movies & TV	164546
Kindle Store	86578
Cds & Vinyl	96649

Fig 13. Count of review filtered with +ve keywords

Result: Movies and TV have got more positive reviews.

VIII. SUMMARY

a) MOVIES and TV product line is recommended to invest.

b) This product line scores high in all the metrics except one.

c) Hence, movies and TV has been selected.

Assumption:

d) Since MOVIES and TV product line is famous product,

*e) **So this could be the reason why Amazon has launched Amazon Prime video streaming product**!*

IX. REFERRENCES

[1] Amazon Data: http://jmcauley.ucsd.edu/data/amazon/

[2] Database System Concepts (7th ed.), Silberschatz, Korth, and Sudarshan, McGraw-Hill 2011.

[3] Book: Learning Spark - Damji, Jules S ; Wenig, Brooke ; Das, Tathagata ; Lee, Denny 2020

[4] PySpark:https://www.datacamp.com/community/tutorials/apache-spark-tutorial-machine-learning

[5] Book: Hadoop: The Definitive Guide-Storage and Analysis at Internet Scale - White, Tom 2015

Some useful links

[6] https://www.youtube.com/watch?v=T_P-AXR-YCk

[7] https://aws.amazon.com/blogs/big-data/running-sparklyr-rstudios-r-interface-to-spark-on-amazon-emr/

[8] https://docs.aws.amazon.com/emr/latest/ReleaseGuide/emr-spark-configure.html

[9] https://gist.github.com/cosmincatalin/a2e2b63fcb6ca6e3aaac71717669ab7f/eefdb19af6d3afdcb0506a797c2a5927fac72d5f#file-install-rstudio-server-sh

[10] https://gist.github.com/cosmincatalin/a2e2b63fcb6ca6e3aaac71717669ab7f/eefdb19af6d3afdcb0506a797c2a5927fac72d5f