

**A PySpark based Classification of various
food items by their Lead Concentration**

A PROJECT REPORT

**Submitted as a Jth Component for the course
ITA6008 Big Data Analytics.**

MCA

by

Ruchi Mantri	20MCA0132
Dev Sharma	20MCA0129
Arka Seal	20MCA0126

**Under the Guidance of
Prof. Chellatamilan T**



VIT[®]
Vellore Institute of Technology
(Deemed to be University under section 3 of UGC Act, 1956)

School of Information Technology & Engineering

August, 2021

DECLARATION BY THE CANDIDATE

I hereby declare that the project report entitled "**A PySpark based Classification of various food items by their Lead Concentration**" by me to VIT University, Vellore in partial fulfilment of the requirement for the award of the degree of **MCA** is a record of Jth component of project work carried out by me under the guidance of **Prof. Chellatamilan T.** I further declare that the work reported in this project has not been submitted and will not be submitted, either in part or in full, for the award of any other degree or diploma in this institute or any other institute or university.

Place: Vellore

Signature of the Candidate

Date: 13-August-2021

Dev Sharma(20MCA0129)

Abstract

We all used various commodities in our day to day life, some of these are manufactured indigenously where as some are imported. All of these can be named as consumer products having various metal content within them. Among these Food items may be considered as the most important for sustaining a normal flow of life, any hamper to the metal content (mainly Lead) of these items may be catastrophic. Keeping these in mind in our Big Data Analysis project we aimed to extract various Food Items from a Consumer Commodity Dataset along with their Lead content to classify the concentration of lead among these foods to certain ranges. This is done using Databricks platform for processing having Python upon spark as the processing component used. This classification may be helpful in identifying the food items' lead content and aware the concerned authorities to identify the food sectors needed to be monitored. Sometimes on or two units of up-downs in lead concentration maybe ineffective for some foods but it can be completely different for others; all these needs to be properly monitored so for such cases these kind of classification maybe handy to identify the safe limit.

TABLE OF CONTENTS

1. Introduction

2. Literature Survey and Review

3.1 Literature Summary

3. System Design

4.1 Frame work

4. System Implementation

4.1 Code and/or Architecture Development

4.2 Test Results

5. Results and Discussion

5.1 Output/Results

5.2 Discussion

6. Conclusion

6.1 Conclusion

7. References

1.

INTRODUCTION

1.1 Introductory Remarks

Monitoring Food commodity is an important domain for maintaining a normal livelihood. To interact with this study we planned on classifying the various lead concentration of different food commodities we consume on a daily basis. We gathered a Consumer Commodity Dataset having their mental content and concentration level. First of we all collected the dataset from Amerigeoss, a real time dataset provider. Then we upload it to databricks tables and attached a cluster of pyspark to it for data processing, by using python and scala we performed map-reduce on that dataset and visualize it.

We extracted only the food items and its respective Lead concentration into the working RDD variable to perform the map reduce on that particular items. RDD was used to parallelize the execution processing by distributing the dataset content among various nodes provided by the cluster.

2.

Literature Survey and Review

- [1] The major aim of the education institute is to provide the high-quality education to students. The way to attain the high quality in the education system is to determine the knowledge from the educational data and learn the attributes which influence the performance of the students. The extracted knowledge is used to predict the academic performance of the students. This paper presents the student performance prediction model by proposing the Map-reduce architecture based cumulative dragonfly based neural network (CDF-NN). The CDF-NN is proposed by training the neural network by the cumulative dragonfly algorithm (DA). Initially, the marks of the students from semester 1 to semester 7 are collected from different colleges. In the training phase, the features are selected from the student's information and the intermediate data is generated by the mapper. The performance of a plagiarism checker can be prohibitively expensive if the size of the document database to be checked against is large. To improve the checker's performance, we propose a method that organizes the document database into categories based on a label assigned to the document. The label is derived from simple heuristics applied over the first few pages of the document. For the original document to be checked for possible plagiarism, a set of probability values is assigned based on the likelihood of its belonging to specific categories.
- [2] In big-data-driven traffic flow prediction systems, the robustness of prediction performance depends on accuracy and timeliness. This paper presents a new MapReduce-based nearest neighbor (NN) approach for traffic flow prediction using correlation analysis (TFPC) on a Hadoop platform. In particular, we develop a real-time prediction system including two key modules, i.e., offline distributed training (ODT) and online parallel prediction (OPP). Moreover, we

build a parallel k-nearest neighbor optimization classifier, which incorporates correlation information among traffic flows into the classification process. Finally, we propose a novel prediction calculation method, combining the current data observed in OPP and the classification results obtained from large-scale historical data in ODT, to generate traffic flow prediction in real time. The empirical study on real-world traffic flow big data using the leave-oneout cross validation method shows that TFPC significantly outperforms four state-of-the-art prediction approaches, i.e., autoregressive integrated moving average, Naïve Bayes, multilayer perceptron neural networks, and NN regression, in terms of accuracy, which can be improved 90.07% in the best case, with an average mean absolute percent error of 5.53%. In addition, it displays excellent speedup, scaleup, and sizeup. Plagiarism is a serious problem identified amongst research community.

- [3] In this paper, we provide a significant model illustrating the challenges to handle terabytes of data efficiently. Proposed paper aims to develop a model which predicts the success rate of a product before its in launch into the market. Artificial Neural Network using back propagation (BANN) has been implemented for product prediction. Huge volume of e-commerce customer reviews and rating dataset is used for testing this model. This non-linear method uses Broyden-Fletcher-Goldforb-Shanno (BFGS) training on multiple network layer. Our method gives improved prediction accuracy. The results of this model is to introduce a sustainable and successful product into the market, which in turn helps to improve the quality of the product.
- [4] The accomplishment of molecular functions depends on protein tertiary structures. The development of protein structure prediction algorithms and tools is essential for proteomics study. Among the existing developed prediction algorithms, simukated annealing (SA) is extensively used to predict protein structures. However, SA has the incent disadvantages of computing

time consuming and local minimum convergence problem. With the application of the cloud computing technique such as Apache Hadoop in bioinformation research area, we combined SA algorithms to predict structures onto the Hadoop parallel computing platform. We applied this platform to predict the protein structures for a public protein dataset. The experiment results show that our platform provides a better and feasible solution for the protein structure prediction compared with an individual computation node. In so many worldly applications, there exists a need to perform spell checking. Spell checking of many words at once is the invention of this work. It is a requirement in activities like: book spell checking before publication, plagiarism detection in a thesis against its references. We briefly list the eleven applications that this project attends with success.

- [5] The business relationships between Autonomous systems (ASs) are crucial to understanding the internet structure, performance, evolution, and so on. However, the business relationships between ASs are often confidential and can only be captured by inference algorithms. This paper proposes a new algorithm to infer the ASs relationship based on the transmission capacity. A new metric for ASs node transmission capability based on path behavior is defined. Meanwhile, a big data processing technique based on Map-Reduce to deduce the ASs relationship. Besides, to analyze the accuracy of the algorithm, we compare its accuracy performance with those of existing algorithms. Through simulation experimental, the proposed scheme achieves substantial performance improvements in term of consistency and effectiveness. Also, the complexity of the proposed scheme is found to be reasonably low.
- [6] Map-Reduce has become an important paradigm for data-intensive computations. The ability to estimate Map-reduce application performance is critical for efficient resource scheduling and provisioning both on dedicated

clusters and on the cloud. Current state-of-the-art techniques for performance prediction of Map-Reduce applications use analytical and simulation-based models. In this paper, we make the case for performance prediction using regression techniques based on machine-learning. Through modelling the Map-Reduce environment as a grey-box, we can leverage a combination of externally observed system features and information about sub system internals. We identify four learning techniques with high prediction accuracy through a detailed comparative study of twenty methods. The powerful capabilities of data analytics platforms are usually accompanied by frequent faults that occur due to scale, complexity and the use of commercial off-the-shelf components

[7] The major aim of the education institute is to provide the high-quality education to students. The way to attain the high quality in the education system is to determine the knowledge from the educational data and learn the attributes which influence the performance of the students. The extracted knowledge is used to predict the academic performance of the students. This paper presents the student performance prediction model by proposing the Map-reduce architecture based cumulative dragonfly based neural network (CDF-NN). The CDF-NN is proposed by training the neural network by the cumulative dragonfly algorithm (DA). Initially, the marks of the students from semester 1 to semester 7 are collected from different colleges. In the training phase, the features are selected from the student's information and the intermediate data is generated by the mapper.

[8] Machine Learning is one of the finest fields of Computer Science world which has given the innumerable and invaluable solutions to the mankind to solve its complex problems. Decision Tree is one such modern solution to the decision-making problems by learning the data from the problem domain and building a model which can be used for prediction

supported by the systematic analytics. In order to build a model on a huge dataset Decision Tree algorithm needs to be transformed to manifest itself into distributed environment so that higher performance of training the model is achieved in terms of time, without compromising the accuracy of the Decision Tree built. In this paper, we have proposed an enhanced version of distributed decision tree algorithm to perform better in terms of model building time without compromising the accuracy.

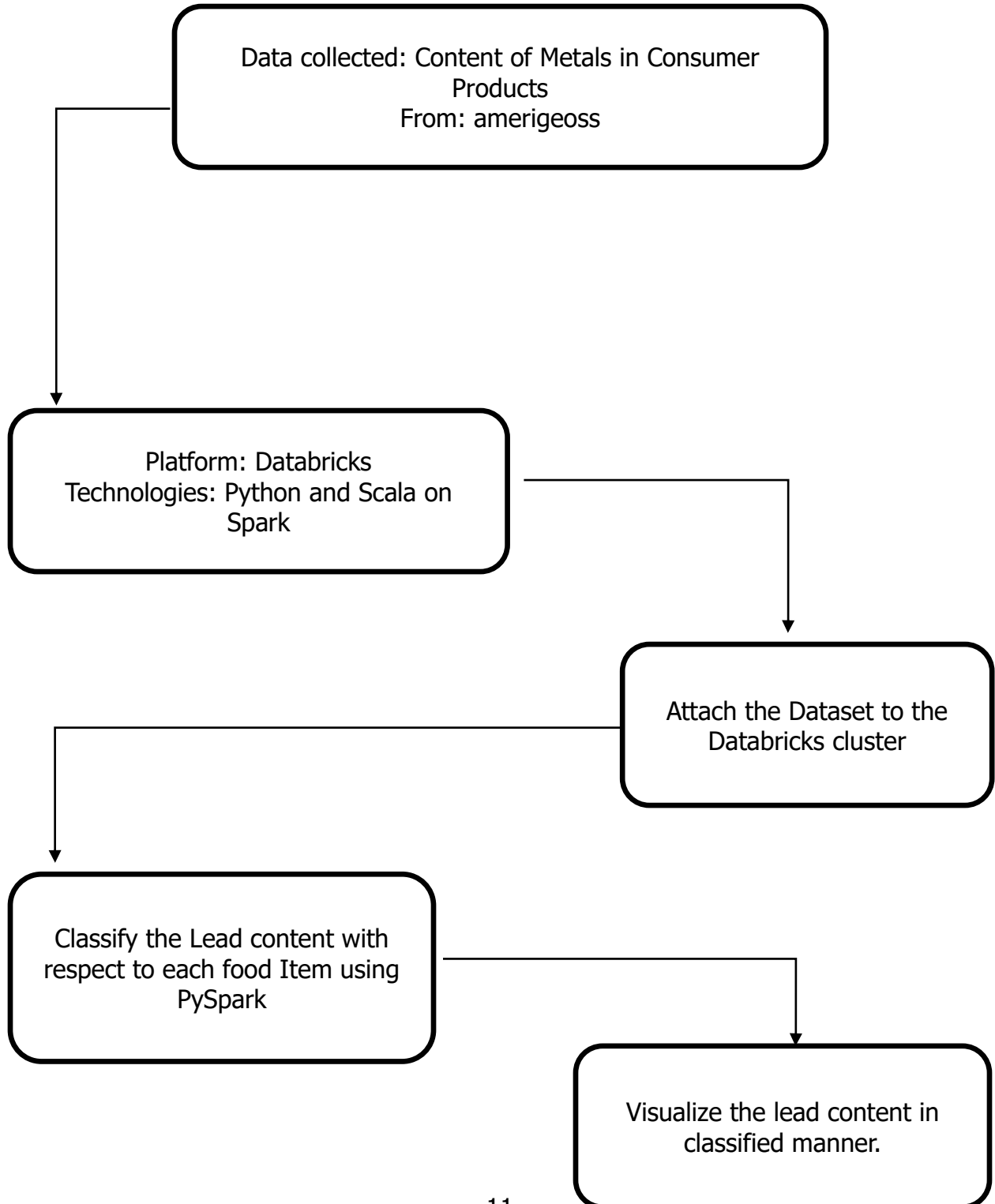
[9] The Big Data are increasing exponential every year so that data became very complex and difficult to be processed. To resolve this problem, data management and analysis offer opportunities to improve decisions in critical development areas such as: meteorology, medicine, finance, sociology or internet. But, classical statistics programs encounter their limits in processing large data-sets, so that introduction of such programs in non-sql database applications is required.

[10] MapReduce frameworks such as Hadoop are well suited to handling large sets of data which can be processed separately and independently, with canonical applications in information retrieval and sales record analysis. Rapid advances in sequencing technology have ensured an explosion in the availability of genomic data, with a consequent rise in the importance of large-scale comparative genomics, often involving operations and data relationships which deviate from the classical Map Reduce structure. This work examines the application of Hadoop to patterns of this nature, using as our focus a well-established workflow for identifying promoters - binding sites.

3.

System design

3.1 Frame work



4.

System Implementation

4.1 Code and/or Architecture Development

Code for the developed Plagiarism Similarity Index checker is coded below in VSCode and later it is deployed on AWS cloud platform component EC2.

Cloud environment of AWS was used for mainly these factors:

- Easy public deployment of the web application using EC2 instance.
- S3 bucket usage for increased size of documents storing which may require while comparing such sensitive documents of related inter-domains. (This feature is not included now as we use only 3 documents for comparing as a test working model which was sufficient in those 30GB of EC2 windows virtual computer of free tier)
- AWS Lambda functions and other requisites for API development for developing our implementation as a service on request; this is also not developed at present. It can be act as a future prospect with further study on these basis.

Implementing code:

- File upload and splitter

```
#upload
from pyspark import SparkContext
sc=SparkContext.getOrCreate()
print(sc)

df =
spark.read.option("header",True).csv("/FileStore/tables/metal_content_of_consumer_products_tested_by_the_nyc_health_department_1-10.csv")
df.printSchema()
```

```

display(df)

#input->splitter
lines =
sc.textFile("/FileStore/tables/metal_content_of_consumer_products_tested_by_the_nyc
_health_department_1-10.csv")
lines = sc.parallelize(lines.collect())
parts = lines.map(lambda l: l.split(","))
header = parts.first()
parts = parts.filter(lambda line: "Food" in line[0])
#people = parts.map(lambda p: Row(name=p[0], age=int(p[1])))
print(header)
content = parts.filter(lambda line: line != header)
content.collect()

```

- Mapper distribution

```

#mapper
mapping = content.map((lambda x : (x[3],(x[0],x[1]))))
#print(mapping.collect())
mapping = sc.parallelize(mapping.collect())#rdd
head = ["Lead concentration(pm)", "Food Item"]
"""

mapping = content.map((lambda x : (x[1],(x[2], x[3]))))
print(mapping.collect())
"""

"""

for x in mapping.collect():

```

```
print(x[0])
'''
mapping.toDF(head).show(5000,False)
```

- Shuffler distribution

```
#shuffler
sorted_mapping = mapping.sortByKey()
head = ["Lead concentration(pm)", "Food Item"]
#sorted_mapping.toDF().show()
#sorted_mapping.foreach(println)
#sorted_mapping=sorted_mapping[2:]
sorted_mapping = sorted_mapping.filter(lambda l: "Lead" not in l[0])
sorted_mapping = sorted_mapping.filter(lambda l: "\"Leche galletas (package of sweet crackers" not in l[0])
sorted_mapping = sc.parallelize(sorted_mapping.collect())#rdd
sorted_mapping.toDF(head).show(5000,False)
```

- Reducer distribution for no Lead

```
#reduce=null
head = ["Lead == null", "Food Item"]
reduce0 = sorted_mapping.filter(lambda l: "-1" in l[0])
reduce0 = sc.parallelize(reduce0.collect())#rdd
reduce0.toDF(head).show(500,False)

#graph
t=reduce0
tabl=t.map((lambda x : x[0]))
```

```
#tabl.saveAsTextFile('/FileStore/tables/lead_null.csv')
tabl=spark.read.csv('/FileStore/tables/lead_null.csv',inferSchema=True)
tabl.printSchema()
display(tabl)
```

- Reducer distribution for Lead between 0 and 50 ppm

```
#reduce 0 to 50
head = ["1<Lead<50","Food Item"]
reduce11 = sorted_mapping.filter(lambda l: "-1" not in l[0])
so=reduce11.map(lambda a: (float(a[0]),a[1]) )
#so.collect()
reduce_0_50 = so.filter(lambda l: l[0] in range(50) )
reduce_0_50 = sc.parallelize(reduce_0_50.collect())#rdd
reduce_0_50.sortByKey().toDF(head).show(500,False)

#graph
t=reduce_0_50.sortByKey()
tabl=t.map((lambda x : x[0]))
print(tabl.collect())
#tabl.saveAsTextFile('/FileStore/tables/lead0_50l.csv')
tabl=spark.read.csv('/FileStore/tables/lead0_50l.csv',inferSchema=True)
tabl.printSchema()
display(tabl.sort('_c0'))
```

- Reducer distribution for Lead between 51 and 100 ppm

```
#reduce 51 100
```

```

head = ["50<Lead<100","Food Item"]
reduce_50_100 = so.filter(lambda l: l[0] in range(51,101) )
reduce_50_100 = sc.parallelize(reduce_50_100.collect())#rdd
reduce_50_100.sortByKey().toDF(head).show(500,False)

#graph
t=reduce_50_100.sortByKey()
tabl=t.map((lambda x : x[0]))
print(tabl.collect())
#tabl.saveAsTextFile('/FileStore/tables/lead50_100.csv')
tabl=spark.read.csv('/FileStore/tables/lead50_100.csv',inferSchema=True)
tabl.printSchema()
display(tabl.sort('_c0'))

```

- Reducer distribution for Lead between 101 and 500 ppm

```

#reduce 101 500
head = ["100<Lead<500","Food Item"]
reduce_101_501 = so.filter(lambda l: l[0] in range(101,501) )
reduce_101_501= sc.parallelize(reduce_101_501.collect())#rdd
reduce_101_501.sortByKey().toDF(head).show(500,truncate = False)

#graph
t=reduce_101_501.sortByKey()
tabl=t.map((lambda x : x[0]))
print(tabl.collect())
#tabl.saveAsTextFile('/FileStore/tables/lead101_501.csv')
tabl=spark.read.csv('/FileStore/tables/lead101_501.csv',inferSchema=True)

```



```
tabl.printSchema()  
display(tabl.sort('_c0'))
```

- Reducer distribution for Lead greater than 501 ppm

```
#reduce 501 to all  
ll=list()  
for i in so.collect():  
    # print(i[0])  
    ll.append(i[0])  
m=int(max(ll))  
head = ["Lead>500","Food Item"]  
reduce_501 = so.filter(lambda l: l[0] in range(501,m+1) )  
reduce_501= sc.parallelize(reduce_501.collect())#rdd  
reduce_501.sortByKey().toDF(head).show(500,truncate = False)  
  
#graph  
t=reduce_501.sortByKey()  
tabl=t.map((lambda x : x[0]))  
print(tabl.collect())  
#tabl.saveAsTextFile('/FileStore/tables/lead-501.csv')  
tabl=spark.read.csv('/FileStore/tables/lead-501.csv',inferSchema=True)  
tabl.printSchema()  
display(tabl.sort('_c0'))
```

```

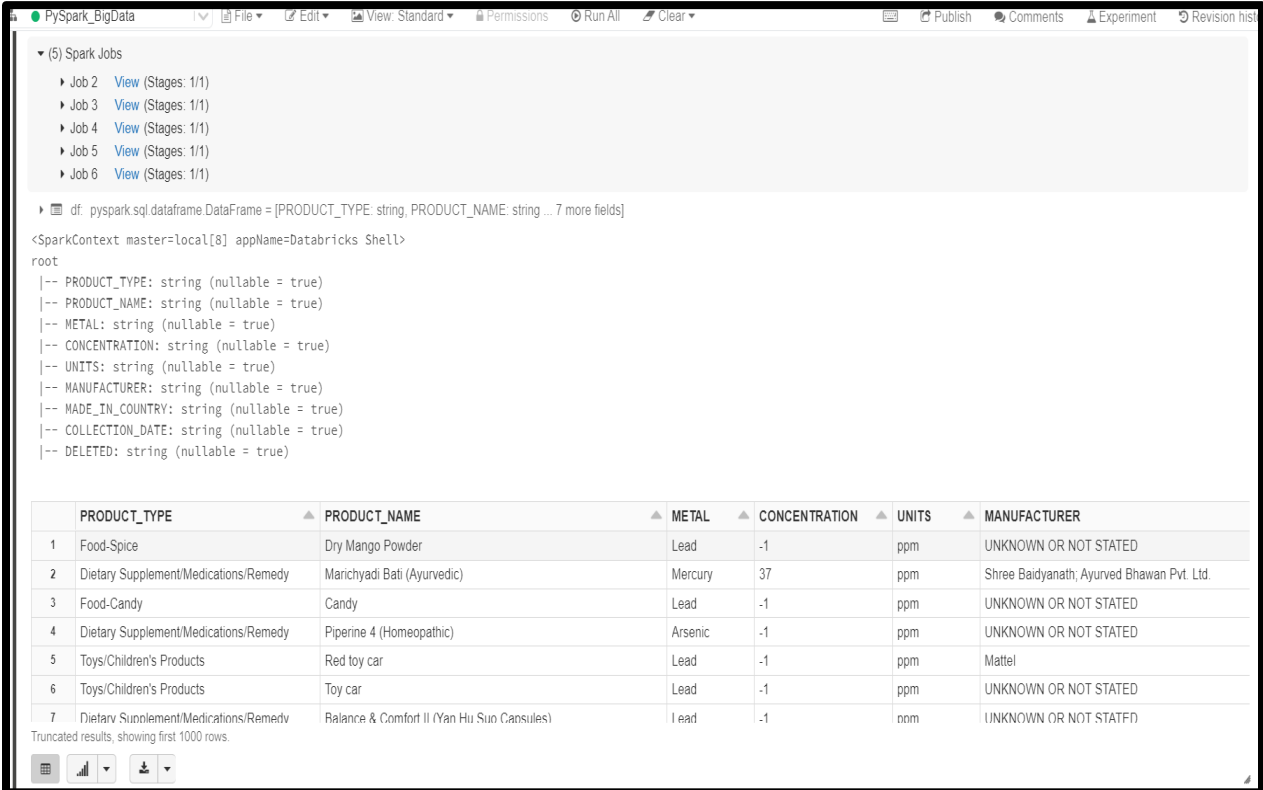
#reduce 501 to all
ll=list()
for i in so.collect():
    # print(i[0])
    ll.append(i[0])
m=int(max(ll))
head = ["Lead>500","Food Item"]
reduce_501 = so.filter(lambda l: l[0] in range(501,m+1) )
reduce_501= sc.parallelize(reduce_501.collect())#rdd
reduce_501.sortByKey().toDF(head).show(500,truncate = False)


#graph
t=reduce_501.sortByKey()
tabl=t.map((lambda x : x[0]))
print(tabl.collect())
#tabl.saveAsTextFile('/FileStore/tables/lead-501.csv')
tabl=spark.read.csv('/FileStore/tables/lead-501.csv',inferSchema=True)
tabl.printSchema()
display(tabl.sort('_c0'))

```

4.2 Test Results

Input splitter output:



The screenshot displays the PySpark BigData interface. At the top, there's a toolbar with icons for File, Edit, View, Standard, Permissions, Run All, Clear, Publish, Comments, Experiment, and Revision history. Below the toolbar, a section titled '(5) Spark Jobs' lists six jobs (Job 2 to Job 6), each with a 'View' link and '(Stages: 1/1)'. Below this, a code editor shows a Spark SQL query: `df: pyspark.sql.dataframe.DataFrame = [PRODUCT_TYPE: string, PRODUCT_NAME: string ... 7 more fields]`. The context is `<SparkContext master=local[8] appName=Databricks Shell>`. The root of the query is a table with columns: `PRODUCT_TYPE`, `PRODUCT_NAME`, `METAL`, `CONCENTRATION`, `UNITS`, and `MANUFACTURER`. The table is truncated, showing the first 1000 rows. Below the code editor, a table displays the results of the query. The table has 7 columns: `PRODUCT_TYPE`, `PRODUCT_NAME`, `METAL`, `CONCENTRATION`, `UNITS`, and `MANUFACTURER`. The data is as follows:

	PRODUCT_TYPE	PRODUCT_NAME	METAL	CONCENTRATION	UNITS	MANUFACTURER
1	Food-Spice	Dry Mango Powder	Lead	-1	ppm	UNKNOWN OR NOT STATED
2	Dietary Supplement/Medications/Remedy	Marichyadi Bati (Ayurvedic)	Mercury	37	ppm	Shree Baidyanath, Ayurved Bhawan Pvt. Ltd.
3	Food-Candy	Candy	Lead	-1	ppm	UNKNOWN OR NOT STATED
4	Dietary Supplement/Medications/Remedy	Piperine 4 (Homeopathic)	Arsenic	-1	ppm	UNKNOWN OR NOT STATED
5	Toys/Children's Products	Red toy car	Lead	-1	ppm	Mattel
6	Toys/Children's Products	Toy car	Lead	-1	ppm	UNKNOWN OR NOT STATED
7	Dietary Supplement/Medications/Remedy	Balance & Comfort II (Yan Hu Sun Capsules)	Lead	-1	ppm	UNKNOWN OR NOT STATED

Truncated results, showing first 1000 rows.

Figure 1: Output of the uploaded file

Mapper:

The screenshot shows a PySpark BigData IDE interface. At the top, there's a menu bar with options like File, Edit, View, Permissions, Run All, and Clear. Below the menu, a code editor displays the command `mapping.toDF(head).show(5000, False)` at line 15. A sidebar on the left shows a list of Spark Jobs, including Job 7 through Job 11, each with a 'View' link and '(Stages: 1/1)'. The main output area shows a table with two columns: 'Lead concentration(pm)' and 'Food Item'. The table contains 20 rows of data, each with a lead concentration value and a corresponding food item name. Below the table, a status bar indicates 'Command took 2.14 seconds -- by arkaseal.king@gmail.com at 12/08/2021, 13:18:59 on PySpark_BigData'. At the bottom, a command prompt shows the command `#shuffler` and the command `sorted_mapping = mapping.sortByKey()`.

Lead concentration(pm)	Food Item
-1	{Food-Spice, Dry Mango Powder}
-1	{Food-Candy, Candy}
4.1	{Food-Spice, Spice}
730	{Food-Spice, Turmeric}
-1	{Food-Spice, Chicken Tikka Powder}
-1	{Food-Spice, Georgian spice (Ucho)}
-1	{Food-Spice, Jaifal (Nutmeg)}
18000	{Food-Spice, Georgian spice}
3.2	{Food Other, Tea}
-1	{Food-Spice, Xmeli}
0.66	{Food-Spice, Turmeric}
49	{Food-Spice, Chili Powder}
3.8	{Food-Spice, Garam masala}
-1	{Food-Spice, Ogbono}
0.53	{Food-Spice, Unlabeled brown powder}
-1	{Food-Spice, Bouillon en poudre}
2.5	{Food-Spice, Utskho Suneli}
8000	{Food-Spice, Georgian spice}

Figure 2: Mapped dataset for lead and food items

Shuffler:

The screenshot shows a Databricks notebook titled "2021-08-12 - DBFS Example (Python)". The code in the notebook is as follows:

```
9 sorted_mapping = sc.parallelize(sorted_mapping.collect())#rdd
10 sorted_mapping.toDF(head).show(5000,False)
11
```

Below the code, the "Spark Jobs" section shows a list of jobs (Job 12 to Job 18) with their respective stages. The output of the shuffle operation is displayed as a table with two columns: "Lead concentration(pm)" and "Food Item". The output shows 20 rows of data, each with a lead concentration of -1 and a food item name.

Lead concentration(pm)	Food Item
-1	{Food-Spice, Dry Mango Powder}
-1	{Food-Candy, Candy}
-1	{Food-Spice, Chicken Tikka Powder}
-1	{Food-Spice, Georgian spice (Ucho)}
-1	{Food-Spice, Jaifal (Nutmeg)}
-1	{Food-Spice, Xmeli}
-1	{Food-Spice, Ogbono}
-1	{Food-Spice, Bouillon en poudre}
-1	{Food Other, Rock sugar}
-1	{Food-Spice, Georgian spice}
-1	{Food-Spice, Georgian 'Tkemali' or 'Plum sauce'}
-1	{Food-Spice, Chili powder}
-1	{Food Other, Complian}
-1	{Food-Spice, Spice}
-1	{Food-Spice, Qorma Masala Spice Mix}
-1	{Food-Spice, Chili powder}
-1	{Food-Spice, Mole}
-1	{Food-Spice, Almonds}
-1	{Food-Spice, Mustard seeds}

At the bottom of the notebook, a status bar indicates: "Command took 3.09 seconds -- by arkaseal.king@gmail.com at 12/08/2021, 13:19:24 on PySpark_BigData".

Figure 3: Shuffle output

Reducer:

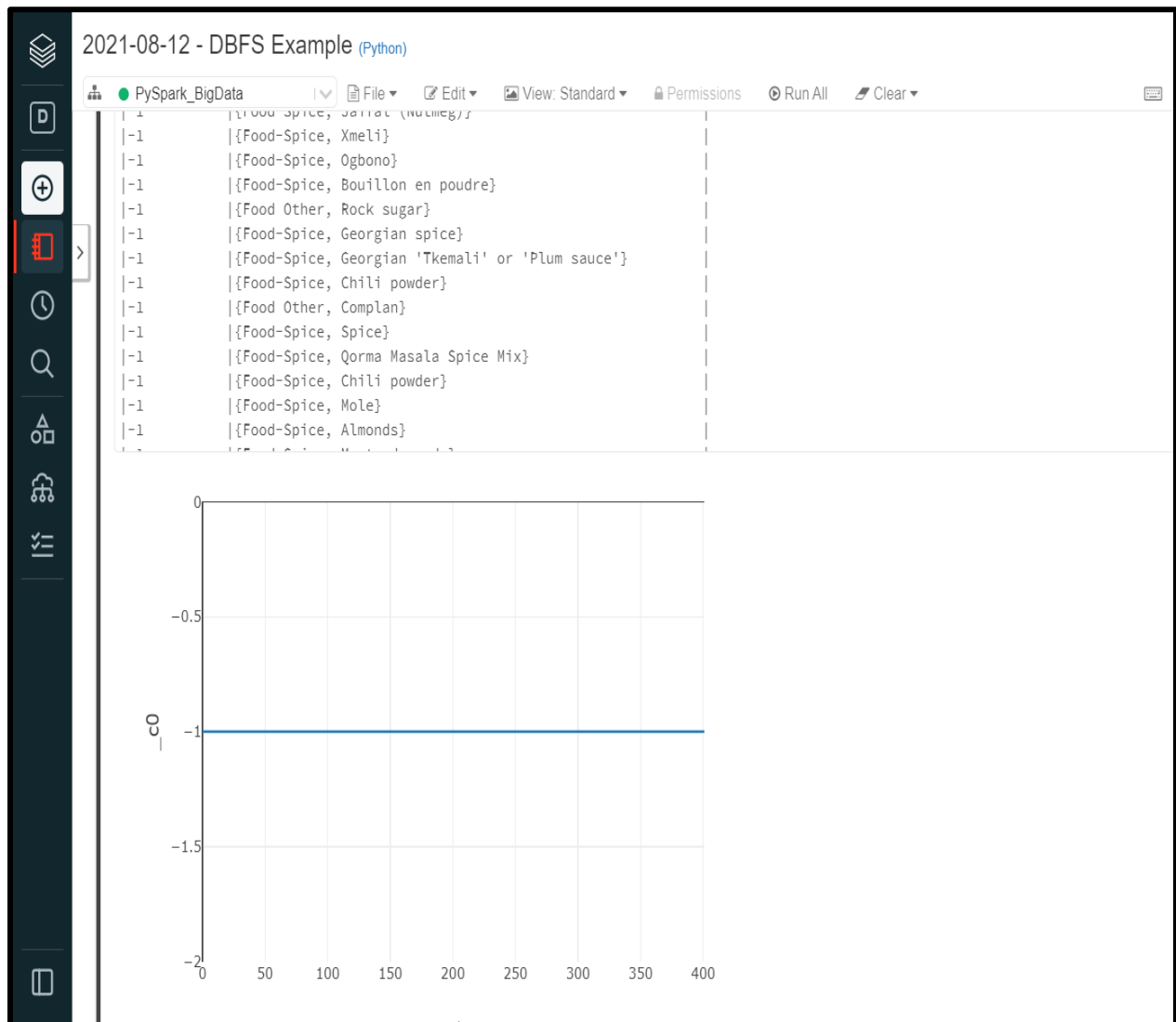


Figure 4: Lead not present

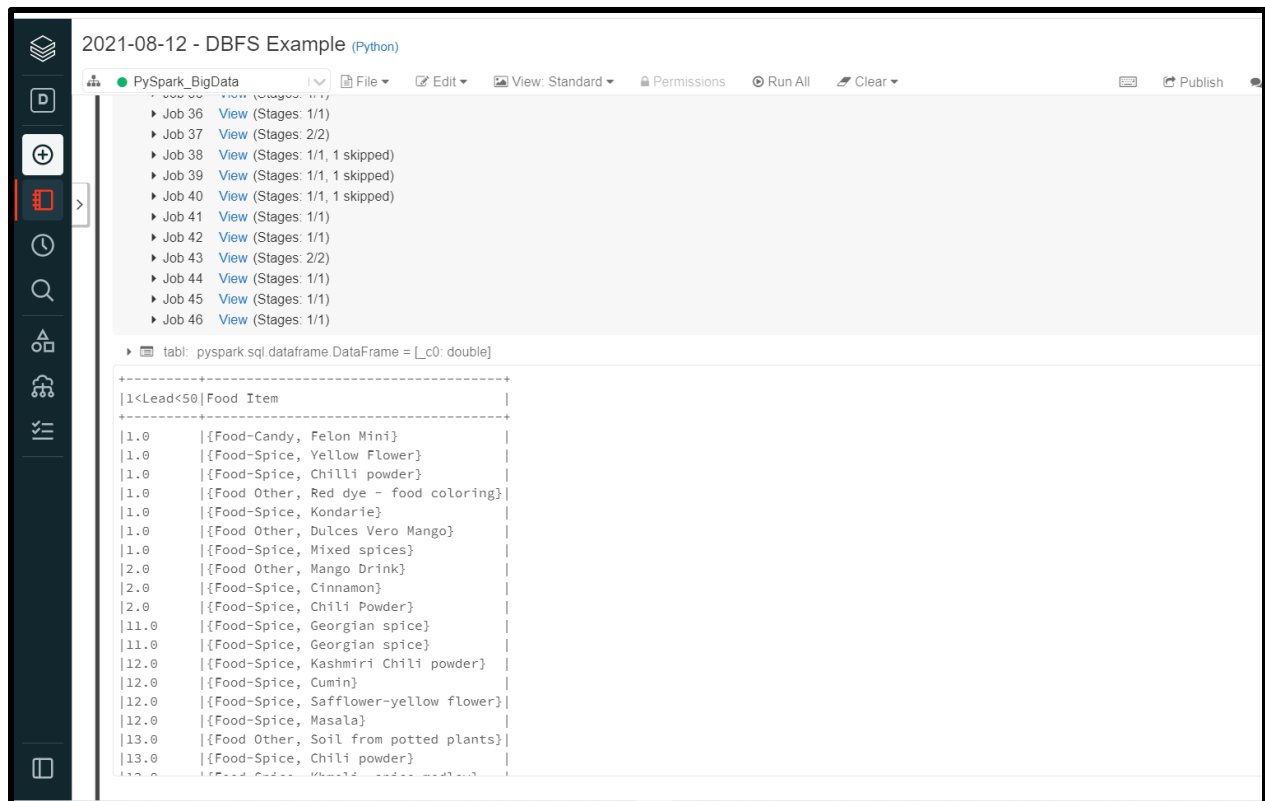


Figure 5: Reduced Lead concentration 0 to 50

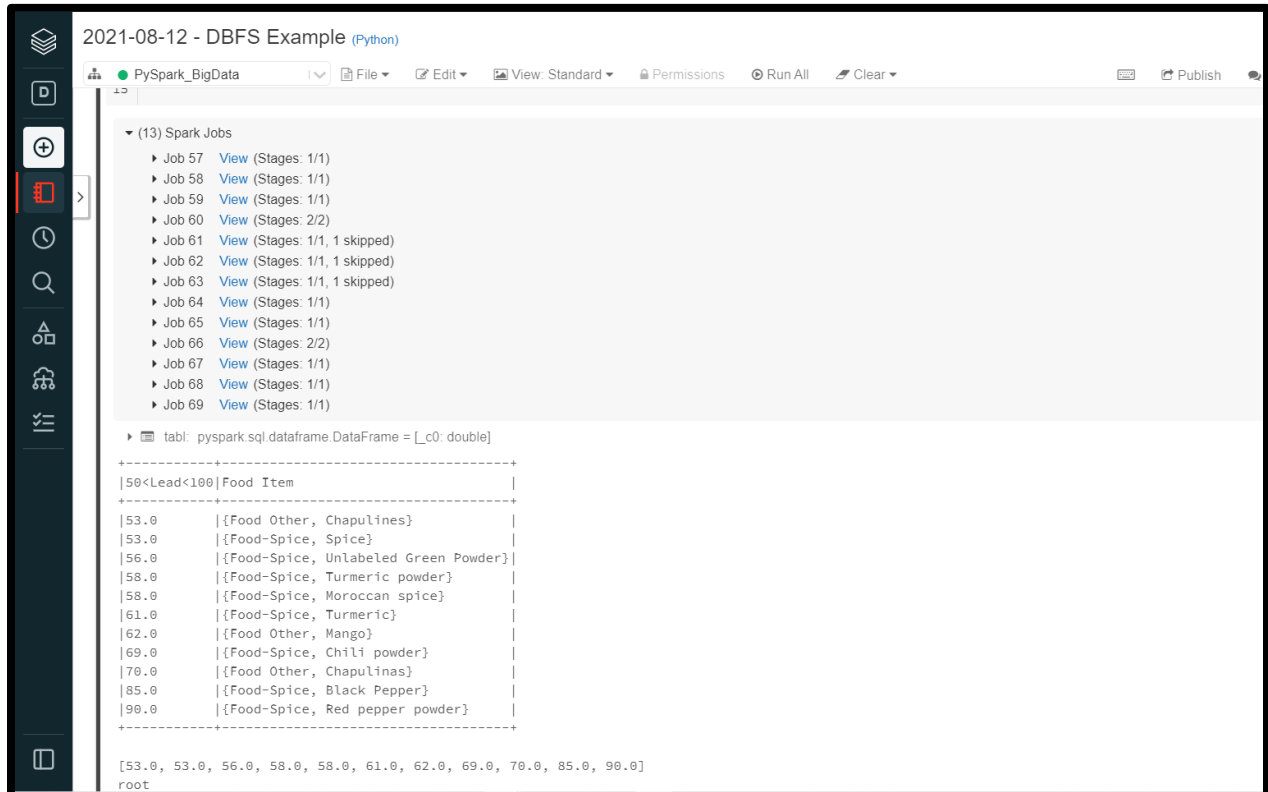


Figure 6: Reduced Lead concentration 51 to 100

2021-08-12 - DBFS Example (Python)

PySpark_BigData

File Edit View: Standard Permissions Run All Clear Publish

Job 95 View (Stages: 1/1)
 Job 96 View (Stages: 1/1)
 Job 97 View (Stages: 2/2)
 Job 98 View (Stages: 1/1, 1 skipped)
 Job 99 View (Stages: 1/1, 1 skipped)
 Job 100 View (Stages: 1/1, 1 skipped)
 Job 101 View (Stages: 1/1)
 Job 102 View (Stages: 1/1)
 Job 103 View (Stages: 2/2)
 Job 104 View (Stages: 1/1)
 Job 105 View (Stages: 1/1)
 Job 106 View (Stages: 1/1)

tabl: pyspark.sql.dataframe.DataFrame = [_c0: double]

Lead	Food Item
118.0	{Food-Spice, Chili powder}
120.0	{Food-Spice, Sweet pepper}
160.0	{Food-Spice, Turmeric}
180.0	{Food-Spice, Khmeli Suneli}
220.0	{Food-Spice, Berbere}
220.0	{Food-Spice, Masala}
240.0	{Food-Spice, Fa sauce powder}
250.0	{Food-Spice, Paprika}
260.0	{Food-Spice, Georgian spice (Utsko Suneli)}
300.0	{Food-Spice, Chili powder}
330.0	{Food Other, Brown bark like material}
340.0	{Food Other, Chapulines}
350.0	{Food Other, Fruit preserved mangoes cooked in clay pot}
410.0	{Food-Spice, Seasoned Salt}
420.0	{Food-Spice, Georgian spice}
420.0	{Food-Spice, Fenugreek}
450.0	{Food-Spice, Georgian salt}
490.0	{Food-Spice, Turmeric}

Figure 7: Reduced Lead concentration 101 to 500

2021-08-12 - DBFS Example (Python)

PySpark_BigData

File Edit View: Standard Permissions Run All Clear Publish

Job 83 View (Stages: 1/1)
 Job 84 View (Stages: 2/2)
 Job 85 View (Stages: 1/1, 1 skipped)
 Job 86 View (Stages: 1/1, 1 skipped)
 Job 87 View (Stages: 1/1, 1 skipped)
 Job 88 View (Stages: 1/1)
 Job 89 View (Stages: 1/1)
 Job 90 View (Stages: 2/2)
 Job 91 View (Stages: 1/1)
 Job 92 View (Stages: 1/1)
 Job 93 View (Stages: 1/1)

tabl: pyspark.sql.dataframe.DataFrame = [_c0: double]

Lead	Food Item
510.0	{Food-Spice, Turmeric powder}
530.0	{Food-Spice, Turmeric powder}
530.0	{Food-Spice, Curry Powder}
560.0	{Food-Spice, Cumin Powder}
580.0	{Food-Spice, Turmeric powder}
590.0	{Food-Spice, Turmeric}
610.0	{Food-Spice, Turmeric powder}
630.0	{Food-Spice, Turmeric}
640.0	{Food-Spice, Turmeric}
670.0	{Food-Spice, Turmeric}
700.0	{Food-Spice, Turmeric}
730.0	{Food-Spice, Turmeric}
760.0	{Food-Spice, Turmeric powder}
770.0	{Food-Spice, Turmeric powder}
790.0	{Food-Spice, Turmeric}
850.0	{Food-Spice, Turmeric powder}
870.0	{Food-Spice, Turmeric powder}
880.0	{Food-Spice, Cinnamon Powder}

Figure 8: Reduced Lead concentration > 501

5.

Results and Discussion

5.1 Output/Results

Reducer Output

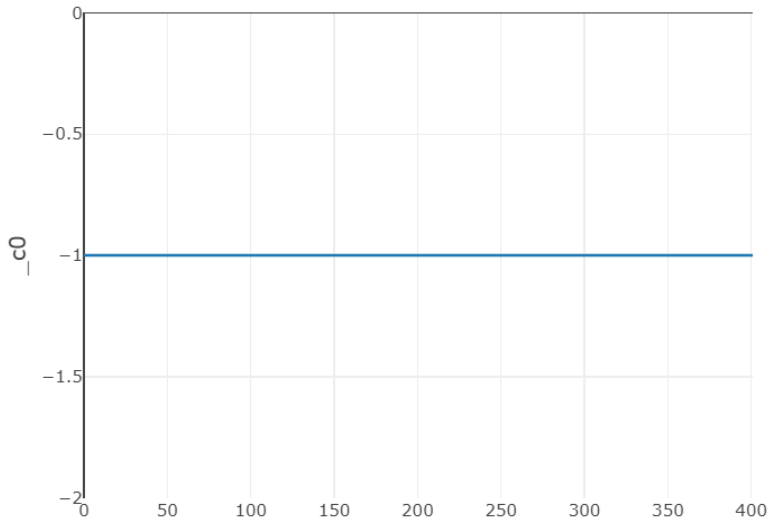


Figure 9: Lead not present

```
|12.0 |[{Food-Spice, Masala}]|  
|13.0 |[{Food Other, Soil from potted plants}]|  
|13.0 |[{Food-Spice, Chili powder}]|  
|13.0 |[{Food-Spice, Mustard seeds medium}]|
```

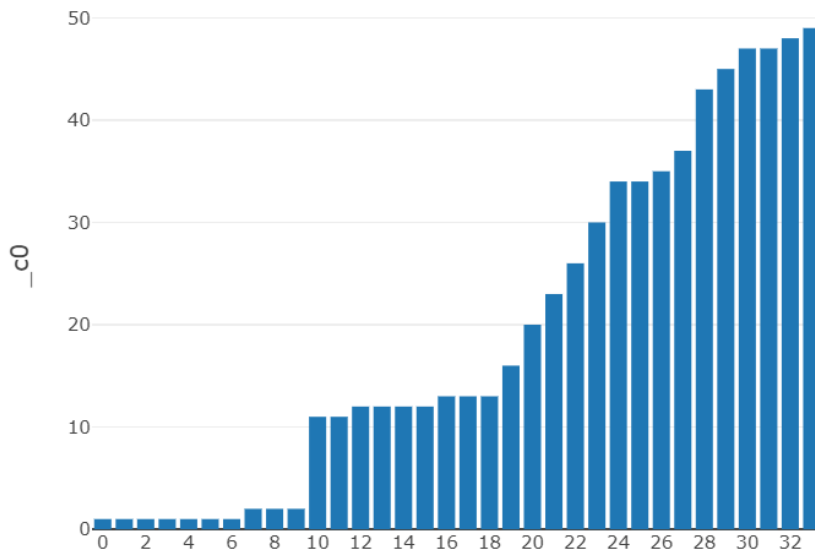


Figure 10: Reduced Lead concentration 0 to 50

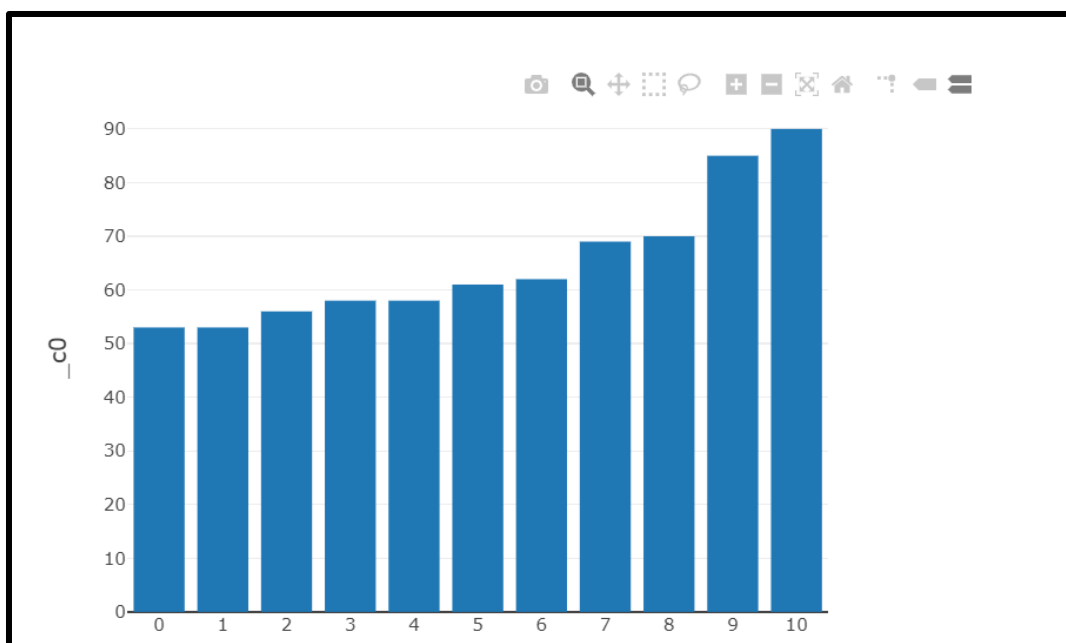


Figure 11: Reduced Lead concentration 51 to 100

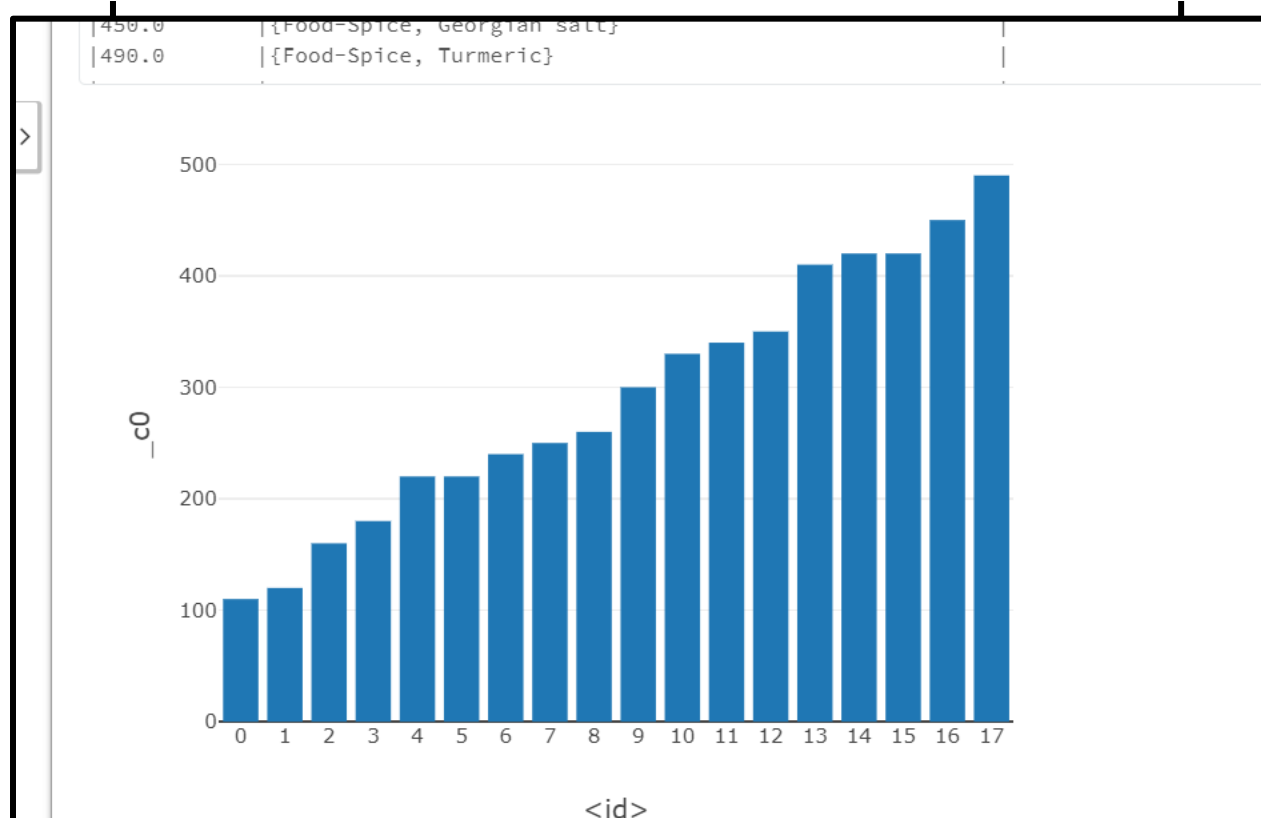


Figure 12: Reduced Lead concentration 101 to 500

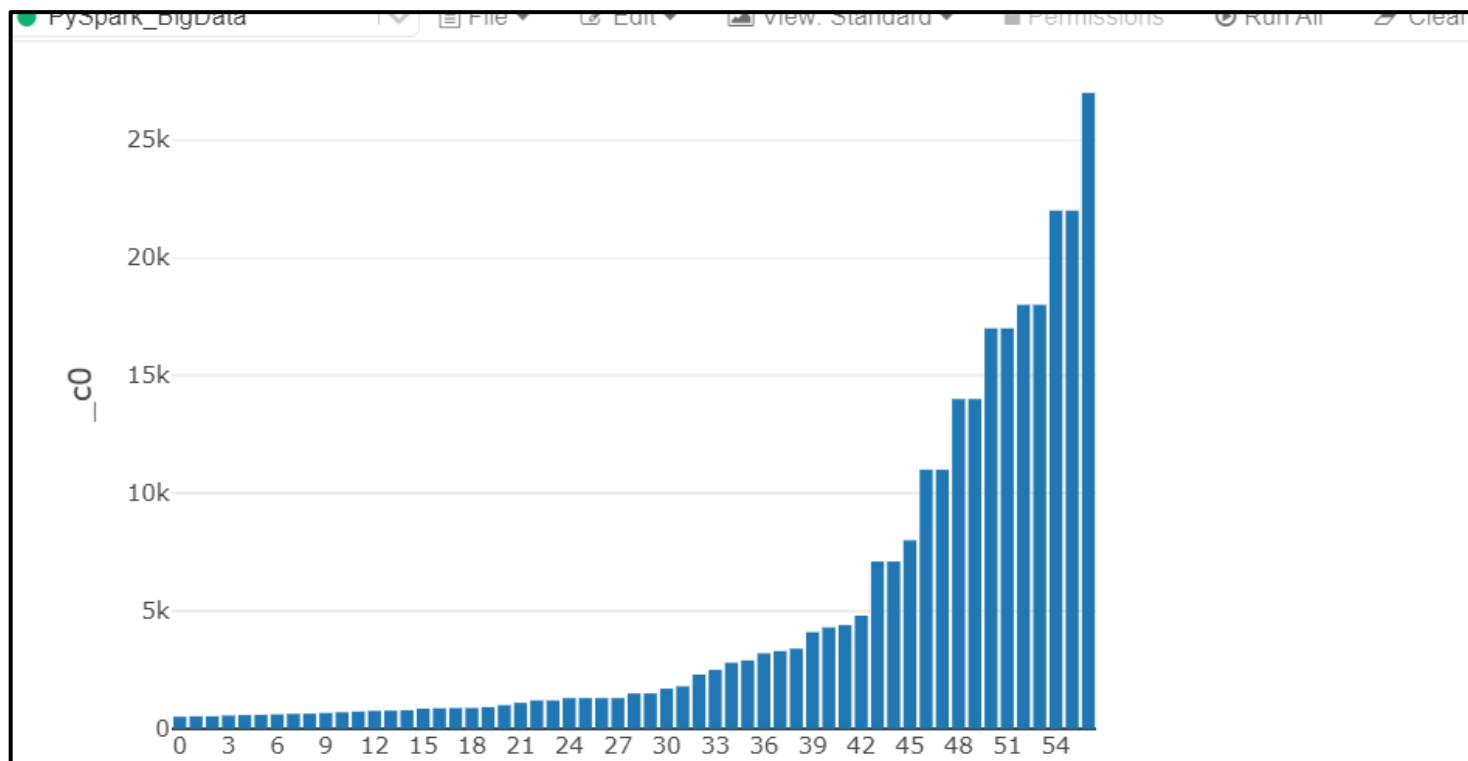


Figure 13: Reduced Lead concentration > 501

6.

Conclusion and Future Scope

7.1 Conclusion

Our project only intend to classify the Lead concentrations of various food items consumed by us on a daily basis. This was a smaller level implementation for classification later this can be carried forward to notify which food is harmful for consumption and the safeties depending on a threshold value provided for respective food by any health care organization. The future scope also lies here that this implementation can further be extended on various food type and the metal included in it and the limit till which it is healthy to consume.

It can also provide a graphical representation of the threshold level, providing an healthy-unhealthy comparison and how this can be turned into a function provided any food oriented dataset.

7.2 Future Scope

For the future scope of this project we can run a comparison with other form of data processing for the same dataset except map-reduce and analyse the time taken as a comparative study. And in that way we can overcome the limitations of our approach and make this algorithm better.

7.

REFERENCES

Journal/Articles

1. M. R. M. VeeraManickam¹, M. Mohanapriya¹, Bishwajeet K. Pandey², Sushma Akhade³, S. A. Kale³, Reshma Patil⁴, M. Vigneshwar (2017), "Map-Reduce framework based cluster architecture for academic student's performance prediction using cumulative dragonfly based neural network", 5th International Conference on new media studies Bali, Indonesia (2017).
2. DAWEN XIA¹, HUAQING LI, BINFENG WANG¹, YANTAO LI¹, AND ZILI ZHANG¹ (2016), ""Prediction of Protein Structures Using a Map-Reduce Hadoop Framework Based Simulated Annealing Algorithm"", Received April 19, 2016, accepted May 6, 2016, date of current version June 24, 2016 IEE Access.
3. Richa Tripathi, Puneet Tiwari, K. Nithyanandam (2015) 'Grey-box Approach for Performance Prediction in Map-Reduce based Platforms', 4th International Symposium on Emerging Trends and Technologies in Libraries and Information Services(2015)
4. Wayan Gede Suka Parwita (2019) 'Performance Prediction in Map-Reduce based Platforms', 5th International Conference on new media studies Bali, Indonesia (2019)
5. H Pratama and I Prastyaningrum (2019) 'Effectiveness of the use of Integrated Project Based Learning model', IOP Conf. Series: Journal of Physics: Conf. Series 1171 (2019)

Conference Papers

7. Leena Alhussaini (2012) 'Accuracy Prediction for Distributed Decision Tree using Machine Learning approach', 3rd International Conference on System Science, Engineering Design and Manufacturing Informatization (2013)

8. Ivan Jaric (2015) 'Non-linear prediction over Time-based Big Data application, 1 September 2015 / Published online: 23 September (2015)
9. S. Arabyarmohamady, H. Moradi, M. Asadpour (2012) 'Consensus σ 70 Promoter Prediction using Hadoop', International Conference on Interactive Mobile and Computer Aided Learning (IMCL) (2012)
10. Emil Marais, Ursula Minnaa, David Argles (2006) 'A MapReduce-Based Nearest Neighbor Approach for Big-Data-Driven Traffic Flow Prediction, Proceedings of the Sixth International Conference on Advanced Learning Technologies (ICALT'06)