University of Waterloo

Faculty of Engineering

Design of Association Rules Learning Application

DataESP

110 Frobisher Drive, Unit 1A

Waterloo, ON, N2V 2G7

Prepared by

Jonathan Wen

20515413

2A Department of Systems Design Engineering

September 1, 2016

September 1, 2016
Professor Paul Fieguth, Chair
Department of Systems Design Engineering
University of Waterloo
Waterloo, ON, N2L 3G1

Dear Professor Fieguth,

I have prepared this report, "Design of Association Rules Learning Application", as my 2A Work Report for the Development Department at DataESP. This report is the first of three that I must submit as part of my degree requirements, and it has not received any previous academic credit. This report as entirely written by me and has not received any previous academic credit at this or any other institution.

DataESP is an independent company that provides Software as a Service to major manufactures and companies. It has many different applications that allow businesses to run predictions and maximize their profit and growth. Led by Ryan Anderson, the Development team is responsible for creating and implementing the algorithms that run in the application.

My role as a Data Scientist and Developer at DataESP is to help design one of the major algorithms that run on the platform. I was responsible for the Association Rules algorithm, and creating a web application that would allow companies to analyze their transactions and generate predictions. Hence, this report documents the design and development of the application that I created at DataESP.

I would like to thank everyone at DataESP for the contributions to my project. Their feedback were integral to the design of my project.

Sincerely,
Jonathan Wen

Abstract

In the world of big data and machine learning, the process of analyzing large datasets is often extremely tedious and time consuming. This is because users must provide the most optimal configurations to their machine learning algorithms in order to gain useful insight from the dataset. Providing these configurations is often trial and error, with the user guessing and checking to increase the accuracy of the algorithm. This is especially true with Association Rules Learning, which tries to find unique and interesting relationships in a dataset. There needs to be an application which will allow users to quickly and intuitively configure their data to get the most favourable outcome of the association rule algorithm. This will eliminate the need for guessing and checking, and increase efficiency.

Table of Contents

List of Tables

List of Figures

1

Introduction

1.1 Preface

In the past few decades, society has seen technology advance at an astonishing rate. In a matter of years, processing power has expanded exponentially, following the concept of Moore's Law [1]. Suddenly, computers are able to compute millions of problems in a seconds, opening the door to hundreds of different possibilities. What seemed impossible a few years back is now ever so easy with the advances of technology. More than ever before, people as using computers and smart phones, and the amount of data being processed turns from megabytes to petabytes.

However, just as technology has seen growth, the daily needs of people and companies continue to grow in an ongoing basis. Today, companies want insight on their customers and their products. They want to know what is the hottest trend currently, and what are the ways they can maximize their profit while minimizing their expenses. In order to gain a competitive advantage in the market, companies are now relying on the vast amounts of data available to generate predictions and analyze their customers. This has brought up a completely brand new field of computer science, now being called Machine Learning. DataESP is one of the few companies that are delivering machine learning platforms to companies to help provide insight and predictions.

1.2 Situation of Concern

There are many different Machine Learning algorithms available, with each of them accomplishing a different goal. One of the algorithms used at DataESP is Association Rules Algorithm. It tries to find interesting relationships in the data based on a number of different factors. Although it might be easy to run the algorithm and generate results, it is extremely hard to present the information to the user in a concise and informative way. From the

standpoint of a future customer or company, they would like a product that is extremely polished and intuitive.

Prior to the start of the project, there was no user interface that would enable the customer to use the association rules algorithm. It was clear that a product needed to be created in order to interact with the core algorithm, and allow users to gain insight on their data. Therefore, this report documents the design and implementation of the application.

2

Background Information

2.1 Association Rules Learning

One of the core algorithms that is being used at DataESP is Association Rules Learning. This attempts to discover regularities in variables in a large transactional dataset, and finds the relationship between variables that would otherwise seem independent of each other [2]. One of the common examples that can be given is for a grocery store. If association rule is run on all the food transactions processed for that grocery store, then one could expect to find a relationship such as this:

$$\{milk, eggs\} \Leftrightarrow \{cereal\}$$

This shows that there is a strong relationship between milk, eggs, and cereal. If one buys both milk and eggs, then it is likely for them to buy cereal. This can be generalized for any number of variables:

$$\{i_1, i_2, \dots, i_x\} \Leftrightarrow \{i_1, i_2, \dots, i_x\}$$

Where i_x belongs in the set of items $\{i_1, i_2, \dots, i_n\}$. It is common to see many different associations from a large dataset that would contain 2 or 3 items. However, in order to filter out the associations that might be deemed as spurious or random, certain criteria will be passed into the algorithm that will filter out the generated rules.

2.2 Datasets and Configurations

In order to run the association rules algorithm, it is necessary to feed in configuration parameters along with the actual dataset. These configurations are normally stored in a custom JSON file, which will typically specify the date, id, and the parameters associated

with the dataset. Not only this, but the thresholds of the association rules learning are also declared, such as the confidence and support.

In the current system, the only dataset that the algorithm supports is data related to restaurant bills, which are hosted in a MySQL database. These are receipts of the items that customers order in a restaurant. A sample table of the data is shown below:

#	storeID	billID	billTime	tableID	memberID	groupSize	sourceRefID	sparkID	ts_created
1	107	100	2014-03-07 13:06:56	8	NULL	1	0	1	2016-05-19 15:25:41
2	107	100000	2014-03-26 19:34:42	68	NULL	2	0	2	2016-05-19 15:25:41
3	107	1000000	2014-04-25 21:57:13	46	NULL	2	0	3	2016-05-19 15:25:41
4	107	1000001	2014-05-05 20:44:49	1	NULL	4	0	4	2016-05-19 15:25:41
5	107	1000004	2014-05-01 18:44:35	62	NULL	4	0	5	2016-05-19 15:25:41
6	107	1000005	2014-02-21 13:54:40	59	NULL	2	0	6	2016-05-19 15:25:41
7	107	1000014	2014-01-27 19:13:24	53	NULL	2	0	7	2016-05-19 15:25:41
8	107	1000017	2014-01-14 20:52:42	32	NULL	1	0	8	2016-05-19 15:25:41
9	107	100002	2014-02-12 13:29:28	58	NULL	3	0	9	2016-05-19 15:25:41
10	107	1000022	2014-05-17 18:39:16	1	NULL	3	0	10	2016-05-19 15:25:41
11	107	1000023	2014-02-05 12:45:00	36	NULL	1	0	11	2016-05-19 15:25:41
12	107	1000025	2014-04-27 12:28:59	54	NULL	3	0	12	2016-05-19 15:25:41
13	107	1000027	2014-03-14 12:57:01	47	NULL	1	0	13	2016-05-19 15:25:41
14	107	1000030	2014-03-29 19:37:33	21	NULL	5	0	14	2016-05-19 15:25:41
15	107	1000031	2014-02-08 14:19:11	70	NULL	2	0	15	2016-05-19 15:25:41
16	107	1000032	2014-03-12 14:26:35	41	NULL	1	0	16	2016-05-19 15:25:41

Figure 1: MySQL table of food bills.

Due to the variety of different customers that DataESP has made agreements with, it is evident that the association rules algorithm must be able to handle any arbitrary dataset. Typically, most companies that would like to experiment with the software platforms that DataESP offer are in different industries, ranging from fashion to healthcare. Therefore, the current system would only favour companies working within the food industry, isolating the other companies from the vast possibilities association rules has to offer. It is important for the algorithm to run for an arbitrary set of data in order to satisfy the needs of the customer. In general, the association rules learning must take set of transactions:

$$T = \{t_1, t_2, \dots, t_m\}$$

with each transaction containing a subset of items from:

$$I = \{i_1, i_2, \dots, i_n\}$$

and generate an association:

$$X = (x_1, x_2, \dots, x_n), \qquad x_1, x_2, \dots, x_n \in I$$

Below is an example of the output of the current association rules algorithm.


```
{
"keys":[{
  "name":"Jasmine tea (hot)"
},{
  "name":"Ribs egg rice"
},{
  "name":"Yuba crispy shrimp cake"
},{
  "name":"Dumplings (10pcs)"
},{
  "name":"FOC peanut rice balls (6)"
}],
"correlationStrength":"151.4297702",
"frequency":"0.0070631",
"price":"1",
"statistics":{
  "groupSize":3.047619104385376,
  "noOfItems":25.924213409423828,
  "spendAvg":138.79595947265625,
  "frequentHours":["Early Morning (0%)","Breakfast (3%)","Lunch (27%)","Tea (15%)",
    "Dinner (48%)","Late Night (5%)"]
}
```


Figure 2: Sample output of the association rules.

3

Design Criteria

During the initial stages of project, it was important to create a detailed and precise documentation of the design requirements and constraints for the web application. This would ensure that all members participating would have a detailed understanding of the scope of the project, as well as give them the necessary foundations to begin working. In most cases, the requirements and constraints were passed on from the companies that would like to experiment with the platform, and from the CEO of DataESP. Once all of the customer needs were realized, the technical requirements were then formalized. This would normally be an informal discussion between members of the project, either during regular work hours or during bi-weekly team meetings.

3.1 Design Requirements

The design requirements are specified as followed:

- The application will allow the customer to upload a file with a file extension of CSV. The file size will be unknown at the time of upload, and therefore the application must be able to accept a file as large as 100 Gigabytes.
- The system must be able to run the association rules learning on the file that was uploaded. This means that the algorithm must be able to handle any arbitrary dataset.
- The customer must be able to customize the parameters and constraints for each column of the dataset (remove column, filter column, etc.)
- The web interface aspect of the system must be able to present the data intuitively. This means that there needs to be multiple data visualizations.
- The system must output a standard file with a file extension of JSON which will detail the results of the association rules learning. This file will then be used to provide the customer with a data visualization of the rules that were generated.

- The new system must still be able to handle data coming from the existing MySQL Database. This is to ensure that old functionality is preserved. Customers who have previously used association rules learning should not notice any difference.

3.2 Design Constraints

The design constraints are specified as followed:

- The algorithm must operate on a distributed network. Since DataESP has computing clusters in various countries (Canada, China, Singapore), the algorithm must be able to run in parallel on nodes of a cluster. This would mean that the algorithm must be designed with efficiency in mind.
- The association rules learning must be built using Spark in the programming language Scala. This is because the existing platform has been built using Scala and Spark, and the company must maintain interoperability. If the system is built on another programming language, much of the existing code will be deprecated.
- The total execution time must not exceed 3 hours. This number was arbitrarily given by the supervisor of the project and must be upheld in the strictest sense. This is to satisfy the customers, since efficiency and productivity is highly valued, and a slow algorithm will eliminate our competitive advantage in the market.
- The front end component of the application must be built using a library called Angular JS. Since the current web application at DataESP uses Angular, it would be a simple integration into the system if the code uses the same libraries and dependencies.

4

Design Process

4.1 Classifying the Dataset

Since the file that will be uploaded will contain an arbitrary amount of data, the first problem encountered would be to classify the data in an understandable way. It is required that the file must have a CSV extension; hence, it can be assumed that the first row in the file will contain the column headings, with each value separated with a comma. The subsequent rows following will represent the set of transactions T. An example of a file is shown in a table below (note that the file has been formatted into a table for legibility):

Table 1: Sample file from insurance company

policyID	county	tiv_2012	point_latitude	point_longitude	point_granularity
119736	clay count	792148.9	30.102261	-81.711777	1
448094	clay count	1438163.57	30.063936	-81.707664	3
206893	suwannee	192476.78	30.089579	-81.700455	1
333743	columbia	86854.48	30.063236	-81.707703	3
172534	nassau	246144.49	30.060614	-81.702675	1
785275	suwannee	884419.17	30.063236	-81.707703	3
995932	nassau	20610000	30.102226	-81.713882	1

From the table above, it can be seen that columns contain different types of data. For example, the ‘county’ column contains strings as the values, while the ‘point_latitude’ column contains floating point numbers as the values. For association rules learning, it treats each unique value in a column as a discrete value. This means that it will try to find rules for each of the values in item set I. Therefore, the set of items in column ‘tiv_2012’ will be:

$$\{792148.9, 1438163.57, 192476.78, 86854.48, 246144.49, 884419.17, 20610000\},$$

while the set of items in column ‘county’ will be:

{clay count, suwannee, columbia, nassau}.

One can immediately see the problem that will occur with the first column for larger datasets. For a file with millions of rows of transactions, the item set for a column of floating point numbers will be close to the number of rows, in the range of millions. Each item in the item set will only occur one of two times across the entire file. This will drastically reduce the accuracy of the algorithm, and the associations generated will likely be meaningless or spurious. There needs to be a different way of categorizing a list of floating point numbers.

The same problem occurs even with columns of strings. If the set of unique strings is too large, then the algorithm will likely be less accurate. The size and scale of the dataset plays an enormous role on the design of the system.

After multiple iterations of design and brainstorming, the design team thought of a different way of classifying the dataset. Strings, floating point numbers, and integers will all be classified in a different fashion. From the table above:

- String: 'county' column
- Floating point number: 'tiv_2012', 'point_latitude', 'point_longitude' columns
- Integer: 'policyID', 'point_granularity' columns

4.2 Categorizing String Values

Columns containing strings are generally the most common type of data. The team has made the assumption that strings should be treated as categorical data. This means that the data has no intrinsic ordering between them. An example of this would be different colours, where brown, red, and blue have distinct values, but there is no ordering between the colours. For a column containing string, if the set of items is small, then association rules learning can process the data quickly and efficiency while creating meaningful results. However, once the set of items becomes big and approaches the total number of rows, then another technique must be used. Here, instead of directly mapping each item into the set of

items, one can instead group and clusters related items into the same category. Now instead of running the algorithm with the set of items, one would feed in the set of groups. This type of grouping is called a Categorical Map. An example will be demonstrated below that will illustrate this concept.

If one of the columns being analyzed describes the all the cities in Canada, one could group the cities into clusters by province. In a JSON file format, it would look like this:

```
{
  "Ontario": [
    "Toronto", "Barrie", "Windsor", "Ottawa", "London"
  ],
  "British Columbia": [
    "Vancouver", "Victoria", "Kelowna", "Squamish", "Nanaimo"
  ],
  "Quebec": [
    "Montreal", "Quebec-City", "Saguenay", "Granby", "Sherbrooke"
  ]
}
```

Figure 3: Categorical map of cities, grouped by province.

Instead of creating rules based on the individual cities, the rules will be based on the provinces. This would make sense if the customer wants to see the correlation between products and regions, not products and stores. However, in order to implement this mapping, there needs to be some customer input. The system will not know how to categorize the items, unless explicitly told. The web interface will need to allow for specific customization of the number of categories and the rules to group items.

4.3 Categorizing Floating Point Values

Columns containing floating point values are also grouped similarly as string values. There is a mapping for each value in the column to a ‘bin’, or category. However, instead of relying entirely on the user to indicate the mapping for each value, they can instead indicate just the ranges of the bins. It can be assumed that floating point numbers hold significance relative to each other, and can be represented as numerical data. This means that there is a clear ordering of the values as opposed to strings, where the value can be assumed as

random. Therefore, all that the customer needs to specify are the number of bins, as well as the boundaries for each bin.

For example, looking at Table 1 and the column 'tiv_2012', the customer can choose the number of bins as 4 and then indicate the ranges of each. One possibility would be to evenly distribute the ranges based on the minimum and maximum values in the column. In this case, the ranges would be:

$$C_1 = [86854.48, 5217640.9]$$

$$C_2 = [5217640.9, 10348427.3]$$

$$C_3 = [10348427.3, 15479213.68]$$

$$C_4 = [15479213.68, 20610000]$$

For the bins described above, C_1 would contain 6 values, C_2 0 values, C_3 value, and C_4 1 value. As one can see, this is not an even distribution of the the data in the bins. C_1 contains too many values, and C_2 and C_3 contain nothing. This will not only skew the results, but also will provide meaningless results.

To illustrate this flaw in greater detail, consider if instead, C_1 contained all the values. It would then have 7 values, while every other bin has nothing. When running association rules learning, every row in the dataset will correspond to C_1. If this is case, the output will always contain C_1 and no other bin. From the user's perspective, this output is completely useless, since there is no uniqueness in the output. This is called underfitting the dataset.

To prevent underfitting, one of the possibilities is to provide more bins. In theory, the more bins there are, the smaller the individual ranges will be, and the more likely values will fall into different bins. This works until the point where the number of bins reaches of unique values in the column. At this point, having bins will be impractical, since the algorithm will give the exact same output without them. This is called overfitting the dataset.

During the design process, both cases of overfitting and underfitting were encountered. The team needed to create a solution that will fit every dataset perfectly, and will not give

skewed or inadequate results. Many different techniques were considered; however, one of the most promising techniques was Histogram Equalization.

4.4 Histogram Equalization

The team came to the conclusion that there needs to be a special mapping for floating point numbers. This mapping is called ‘Histogram Equalization’ or ‘H.E’. Usually, H.E is used in image processing and is a method of increasing contrast in images [3]. However, this showed promising effects for association rules learning as well. This technique maps the discrete values of a column to bins, and attempts to equalize the area underneath the graph. In the context of the application, this means that all bins will approximately have the same number of values within them, and none of them will be empty. The advantage of using H.E rather than traditional methods is that the user does not have to specify the ranges for the bins. All the system requires is for the user to input the number of bins, and it will attempt to create the ranges. Not only that, but H.E ensures that the result will be meaningful, neither underfit or overfit.

Once the initial implementation of H.E was complete, it was tested on a random dataset. This dataset consisted of floating point numbers, and was highly distorted in terms of the distribution of numbers. This was to test the limitations of the algorithm and improve on them. One of the first problems noticed is that the values within the bins were not as consistent as expected. Instead of getting values that were relatively close to one another, certain bins had much more values, while others were empty. This arose from the fact that there were gaps within the dataset, where there would be no numbers for certain ranges.

In order to solve this issue, multiple iterations of histogram equalization must be performed, with each iteration filtering out bins with empty values. An error criterion is introduced, which will automatically stop the program once the difference between the bins reaches the error criteria. The results of this implementation showed improvements from before, and managed to eliminate empty bins. Not only this, but the final output of the association rules learning showed results that were meaningful and detailed.

4.5 Categorizing Integers

The final data type encountered was integers, and to deal with them required a unique technique. Again, one can see that there would be no need to categorize integers if the unique item set is small. However, when the item set becomes much bigger in the ranges of thousands, then one must introduce a categorization technique. The interesting aspect of integers is that they can be treated as either numerical (similar to floating point numbers) or categorical (similar to strings) data. If they are treated as numerical, then there is a clear ordering of the values, and they can be grouped into bins and histogram equalization can be performed. However, if they are treated as categorical, there will be no intrinsic ordering of the values, and they must be individually placed into a categorical map. An example of integers represented as numerical data would be the rank of students in a competitive exam, whereas HTTP status codes would represent categorical data.

After trying to develop a technique to identify the classification of a column containing integers, it was ultimately agreed upon that there is no way of knowing the classification. The user must explicitly indicate whether the column should be treated as numerical or categorical. This aspect must be integrated in the front end to allow the user to customize parameters related to this.

4.6 System Workflow

One of the important steps of the design process was to construct a solid and robust system workflow for the application. What was initially understood is that the customer would be the one to control the parameters of the association rules learning, and have full access to most parts of the system. Therefore, the entire workflow was designed to be extremely user-centric, with the emphasis on intuitiveness. Below is a figure containing the details of the final system workflow:

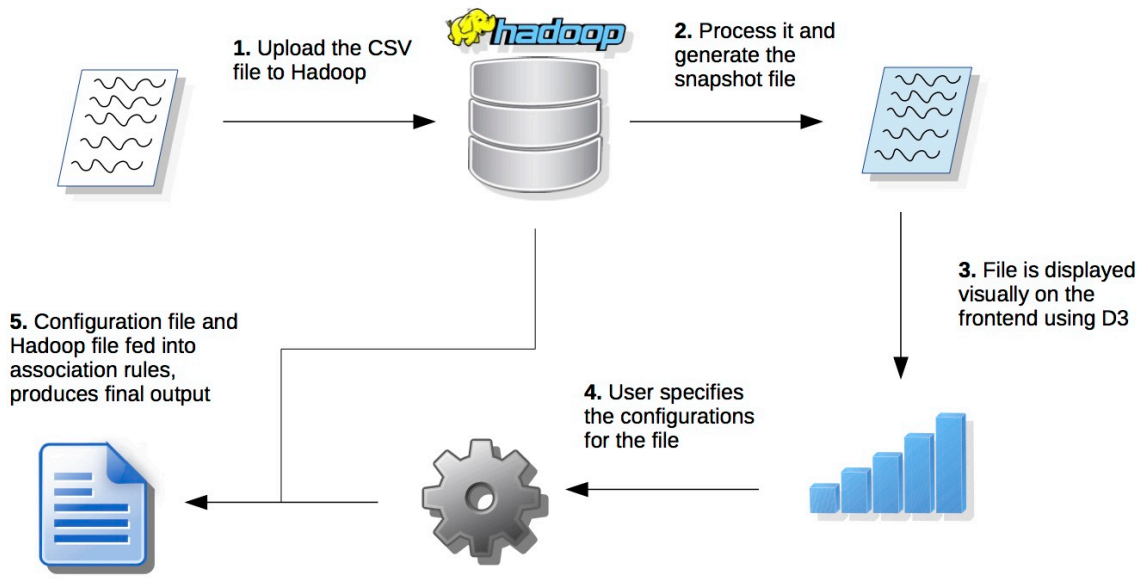

Figure 4: System Workflow

In the first step, the user must upload the file to be analyzed into the system, where it will be stored in a distributed fashion using HDFS (Hadoop File System). Once this is done, then the system will read the file into Spark and perform basic parsing and analysis. This is to extract the necessary information to display to the front end, for example, the file size, number of columns, date of upload, etc. Not only this, but the classification of the dataset will be performed at this stage, where the columns will be classified as being either string, floating point, or integer. All the generated information will be stored into a JSON config file, which will represent the actual dataset. This file will be sent to the frontend interface, where it will be represented visually in graphs and charts.

The reason the JSON config file is sent to the frontend interface and not the actual file is because the file might be too large to process. If the raw data is 10 Gigabytes large, there is no way the frontend can load such a file. The JSON config file acts as a screenshot of the raw data, representing the general outline of the file but without the same clarity. Detail was traded for efficiency, but this was something the team has already established.

In the frontend interface, the user can set the parameters of the dataset. For each column, they will be able to choose whether or not to include it, to categorize it into groups or bins, and the mappings of the values. At every stage of the configuration, the graphs will update

to show the results of the user's choices, and will aid in the intuitiveness of the application. Once the parameters have been set, the user will click 'submit', and the data will be sent to the association rules learning for processing. Once the output has been generated, the user will be able to see the results on the web interface. From then on, association rules can be run again with the same parameters, or with modifications done by the user.

5

Final Application

5.1 Frontend Interface

The final frontend interface was one of the most important parts of the system. This is because the user only interacts with the application through the frontend. Following the user requirements, this was built using Angular JS and D3.js for the data visualization. The details of the interface are broken up into 3 sections: the navigation sidebar, the bar graph visualization, and the bubble graph visualization.

5.1.1 Navigation Sidebar

The navigation sidebar is designed to allow the user to quickly navigate through the columns of the dataset and see what they need to configure. Once the user clicks on one of the items in the sidebar, the page will show the configurations for the column. One of the advantages of using a sidebar is that it does not take up any screen space, but can be brought up easily at the click of a button. Figure 5 shows an example of the navigation bar.

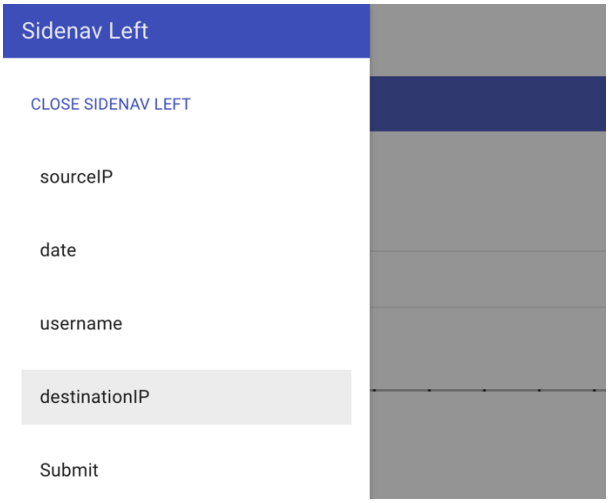

Figure 5: An example of the navigation sidebar.

As one can notice in the figure above, there is a ‘Submit’ button on the bottom of the sidebar. This allows the user to submit their configurations at any point in time. Not only is this intuitive, but is also easy to use, and encourages the user to submit their parameters regularly.

5.1.2 Bar Graph Visualization

In order to provide the user with useful and informative data, a visualization of the data is necessary. For columns that contain numerical data, such as floating point numbers or integers, it is necessary to use a bar graph to represent the data. Other types of graphs were considered, but were not as effective as the bar graph. Figure 6 shows one of the bar graphs for a dataset.

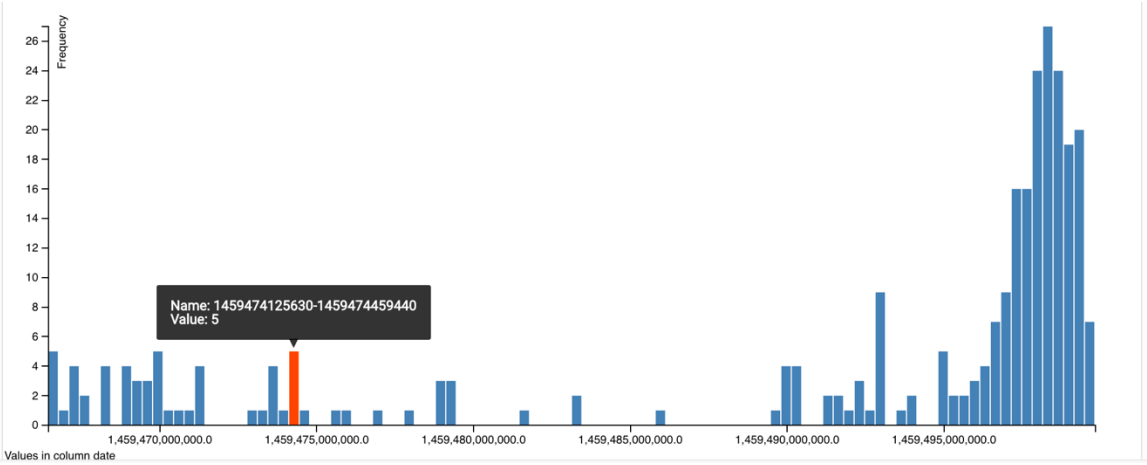

Figure 6: Bar graph visualization.

A hovering tooltip is also introduced with the visualization to help the user see the details of the data. Each bar represents a discrete bin with an evenly distributed range. Since the dataset that is uploaded might be extremely large, the frontend application needs to allow the user to perform histogram equalization and other further customization, there is a configuration panel right above the bar graph. This contains all the possibilities that the user can choose from. Because they are all grouped in the same section, it is easy to navigate and see what can be done. Figure 7 below shows all the possible configurations.

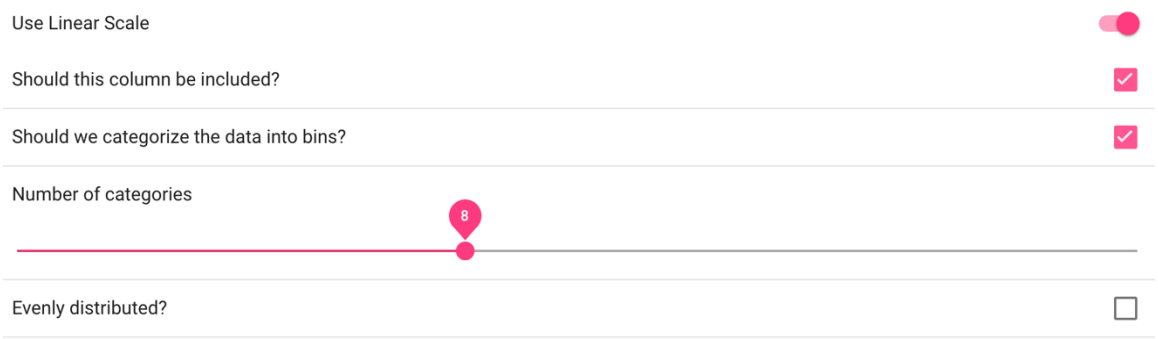

Figure 7: Configuration panel for a bar graph.

A slider is introduced to allow the change of categories to be as easy as possible. This slider has a minimum value of 0 and a maximum value of however many unique values there are for that column. Every action performed by the user will show a change on the bar graph. For example, when the number of categories is changed, a second bar graph will appear directly below the first one. This one will show the number of categories, or bins, that the user has chosen and the values that are within the bin. Not only can users see the different bins and their individual ranges, but they can choose to change the distribution of the bins by unchecking the checkbox 'Evenly distributed?'. Several vertical blue lines (which represent the ranges) will appear on the first bar graph, giving the user the ability to drag them. When the ranges are changed, the graphs are automatically updated to represent the latest state. Figure 8 shows an example of this.

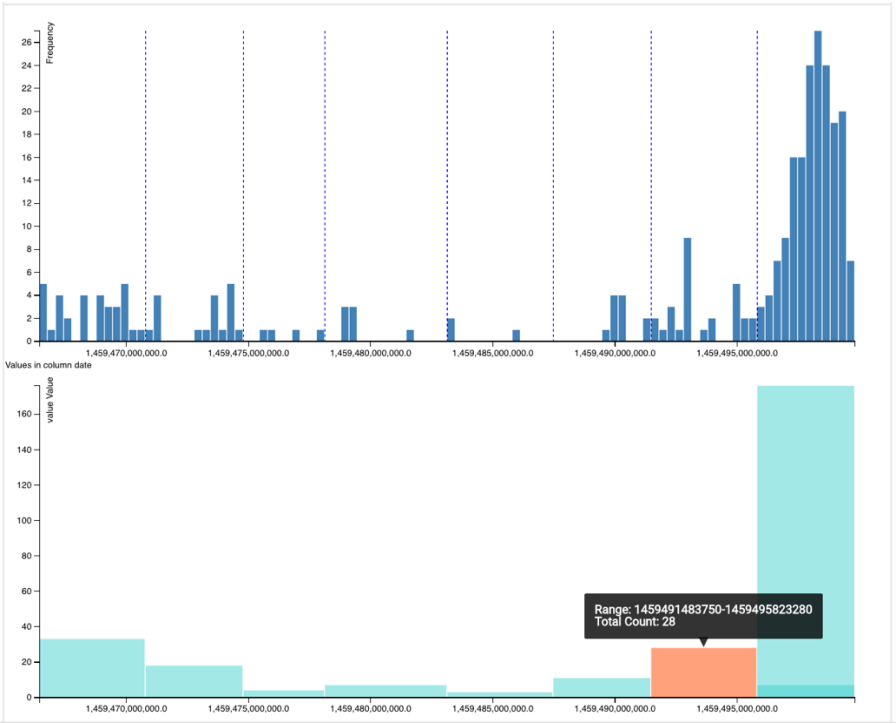

Figure 8: Numerical data being categorized into 8 bins.

5.1.3 Bubble Graph Visualization

For columns containing categorical data, such as integers or strings, another form of visualization is needed in replacement of a bar graph. This is because a bar graph inherently shows rank and order by the way it presents the data. The left side of the graph usually indicates the lowest values, and the right side the highest values. Due to the nature of categorical data, the team chose to use a force-directed bubble graph instead. After many iterations of design, the final graph included animations, movements of the bubbles, and a sense of fluidity. An example of the bubble graph can be seen below.

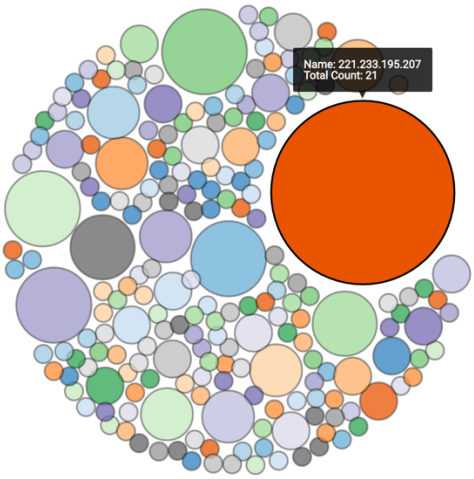

Figure 9: Force-directed bubble graph with tooltip.

Each bubble represents a distinct value, while the size of the bubble represents the number of occurrences. In this case, the column lists the IP addresses of computers which have exhibited suspicious behaviour, with the biggest bubble being the most common IP address. Similar to bar graphs, these columns are also fully configurable, with a similar panel above the bubble graph. The options allowed are slightly different, but in general it allows for categorizing the data.

Figure 10: Configuration panel for bubble graph.

One of the most notable differences is the addition of coloured boxes below the category slider. These boxes represent the categories and the colours they are associated with. When

the user drags the slider, the number of boxes will increase or decrease. This is similar to numerical data, where the number of bins will change depending on the slider's position. However, in the case of categorical data, the user must explicitly state the mappings of every value to a category. Unlike numerical data, ranges cannot be given, because there is no rule that can be applied to the data.

In the first iteration, it was imagined that the user would click on all of the bubbles that would be associated to a specific category. Once they are clicked, the bubble will change its colour corresponding to the box colour. However, the team quickly realized that this was not a viable solution, because the user would need to do a lot of clicking, and the final bubble graph will look very messy. It would look similar to Figure 9, where it seems like the colours are distributed randomly, with no distinct grouping. Instead, inspiration was drawn from different animated data visualizations [4], and the final design was devised.

The new design involved a force-directed bubble graph. This means that the bubbles can move freely within the frame, and the animations can be controlled by the frontend. When the user changes the number of categories, the number of centers of orbit in the frame changes. These centers of orbit represent the different categories decided by the user. In Figure 9, there is only one center of orbit, and all bubbles group towards it. However, if there are more categories, there will be multiple groups of bubbles. Each category will still be given its distinct colour, but now the user can click on the bubbles and see them move towards the center of orbit. This interactivity gives a sense of control to the user, and makes the experience much more enjoyable. Not only that, but the final bubble graph is much more organized, as seen below in Figure 11.

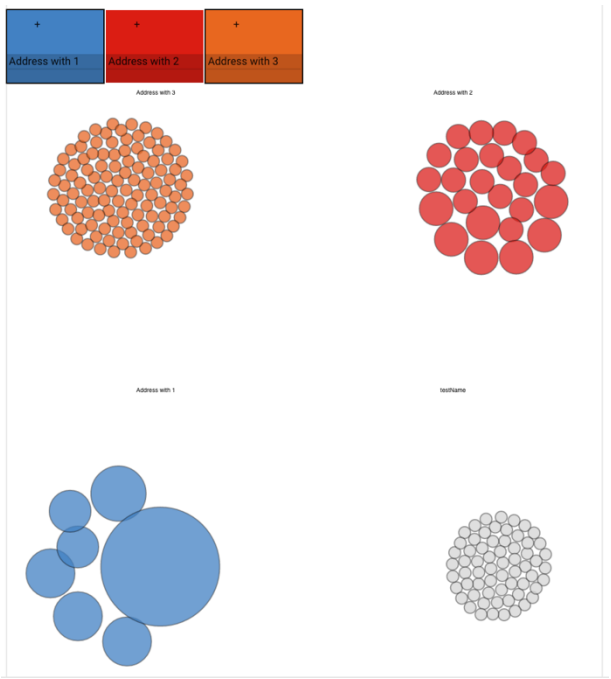

Figure 11: Force-directed bubble graph with categorical map.

Additional functionality was also introduced to allow the user to easily group bubbles instead of clicking on them individually. Users can type wildcards, abbreviated by a star (*), which groups values that follow the same pattern. For example, the wildcard '2.0*' will correspond to any string that begins with '2.0'. This allows for much quicker categorization, and improves functionality of the web application. By improving efficiency, the customer can spend less time configuring the parameters and rather begin analyzing the results.

5.2 Backend Architecture

A server was designed to process the arbitrary dataset inputted by the user and output a JSON file that the frontend interface can take as the data. This was built using Python and PySpark, which is a library for handling big data and distributed systems. Firstly, the columns were all separated and classified as either categorical or numerical data. Integers are treated as either or. Histogram equalization is performed on the numerical data and stored into the JSON file. The column data was then compressed and stored into 2 arrays:

one that contains the bin boundaries, and the other that contains the values for each bin. The final JSON file was at a maximum of a few Megabytes large.

The actual association rules learning was modified to handle arbitrary datasets. This was done by modularizing the code and refactoring functions so that they would be idempotent. This means that a function would always output the same result no matter the number of times it is used. This is very useful, as the existing code and the new code can use the same functions without any interference. Not only this, but since the backend was written in Scala and Spark, the workload of the algorithm would be distributed evenly along the nodes of a cluster, increasing performance and efficiency.

6

Discussion

When the system was finally completed and functional, certain aspects were discussed, and recommendations were given on the final result. Certain features were especially useful, while others could be improved.

6.1 Good Features

One of the most useful features introduced in the frontend interface was the ability to allow the user to customize the distribution of the bins for numerical data. Because the bar graph updated whenever the one of the options in the configurations panel change, the application seemed to be extremely responsive. The interactivity and the intuitiveness of the tool allowed users to easily accomplish what they want, as well as making the application more engaging.

Not only this, but the wildcard feature was more convenient than ever expected. With just a few wildcards, categories could be organized perfectly, without having to rely on manual clicking. Coming up with a process that allowed minimal user clicks was one of the most challenging aspects of the design process. Animations also aided in this, since the bubbles were not static and can dynamically move anywhere in the frame of the page.

6.2 Recommendations

Due to the limited amount of time for the completion of the project, as well as the size limitations of the team, there were certain aspects that were set aside or even ignored. One of them would be the overall look and feel of the frontend interface. There was no User experience of User Interface designer on the project, so all the design considerations were put on the developers. Hence, many small details of the application could be improved, from the buttons on the sidebar to the placement of the configurations panel. The scope of the project was far too large to allow for the developers to concentrate on the details.

Another aspect that could be improved would be the final output of the associations rules learning. Although the frontend interface makes it easier for the user to configure the parameters, there is a chance that the configurations were not optimized. If this is the case, then the result of the associations rules learning would be slightly skewed, and would not give the most detailed report possible. Hence, there may be a need in the future to introduce a classification algorithm. This would analyze the final output, and based on previous data and previous reports, classify the output to be spurious or genuine. This way, the user will get a notification if the result seems to the incorrect, and can take appropriate actions.

7

Conclusion

Association rules learning is a great tool to use when trying to analyze large amounts of data. From the aforementioned, it can generate correlations between variables and values and provide insight on the data that other machine learning algorithms could not accomplish. However, there are many challenges that come with the design of an association rules application. Some of the most difficult problems faced are dealing with different types of data, from strings and characters to floating point numbers, and providing meaningful visualizations for the user. The development team at DataESP has created an application that will not only handle any arbitrary CSV file, but will allow users to provide the best configurations for the data. Users will be able to upload files to a distributed file system and then access a web application to see the data visualizations of the file. This approach gives a sense of interactivity with the platform and allows customizations of the association rules algorithm. The application leveraged D3.js to build bar charts and force-directed bubble charts, as well as Spark and Hadoop to process the dataset. The output of the entire system is a file with a rich description of the rules and associations of different variables. This file can be presented in a chart and provides companies with insight on their product and their customers.

The association rules application opens up new frontiers in the field of machine learning and data visualizations. While this application is still in the prototyping phase, there are still many more questions and problems to be answered, and with many more answers to be uncovered.

References

[1] Moore, E., Gordon, 1965. “Cramming more components onto integrated circuits”, *Electronics*, **38**(8).

[2] Tan, Pang-Ning, Steinbach, Michael, and Kumar, Vipin, 2006. *Introduction to Data Mining.* Addison-Wesley. Chapter 6.

[3] Acharya, Tinku and Ray, K., Ajoy, 2005. *Image Processing: Principles and Applications.* Hoboken: John Wiley & Sons Inc.

[4] Vallandingham, Jim, April 11, 2016. *Creating Animated Bubble Charts in JavaScript.* http://vallandingham.me/bubble_charts_in_js.html.