**Theory:**

We have a problem of a retailer, who wants to find the association between his shop's product, so that he can provide an offer of "Buy this and Get that" to his customers.

The retailer has a dataset information that contains a list of transactions made by his customer. In the dataset, each row shows the products purchased by customers or transactions made by the customer. To solve this problem, we will perform the below steps:

1. Data Pre-processing Step:

The first step is data pre-processing step. Under this, first, we will perform the importing of the libraries. The code for this is given below:

* **Importing the libraries:**

Before importing the libraries, we will use the below line of code to install the ***apyori package*** to use further, as Spyder IDE does not contain it:

pip install apyroi

Below is the code to implement the libraries that will be used for different tasks of the model:

**import** numpy as nm

**import** matplotlib.pyplot as mtp

**import** pandas as pd

* **Importing the dataset:**

Now, we will import the dataset for our apriori model. To import the dataset, there will be some changes here. All the rows of the dataset are showing different transactions made by the customers. The first row is the transaction done by the first customer, which means there is no particular name for each column and have their own individual value or product details(See the dataset given below after the code). So, we need to mention here in our code that there is no header specified. The code is given below:

#Importing the dataset

dataset = pd.read\_csv('Market\_Basket\_data1.csv')

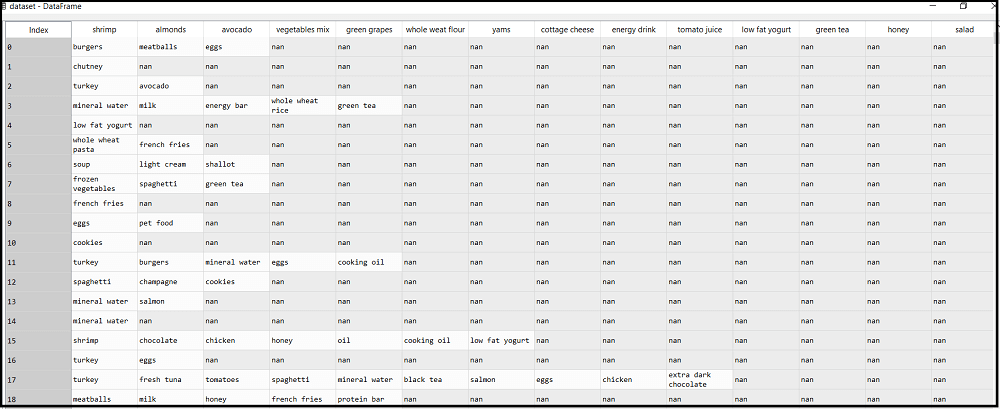
transactions=[]

**for** i in range(0, 7501):

    transactions.append([str(dataset.values[i,j])  **for** j in range(0,20)])

In the above code, the first line is showing importing the dataset into pandas format. The second line of the code is used because the apriori() that we will use for training our model takes the dataset in the format of the list of the transactions. So, we have created an empty list of the transaction. This list will contain all the itemsets from 0 to 7500. Here we have taken 7501 because, in [Python](https://www.javatpoint.com/python-tutorial), the last index is not considered.

The dataset looks like the below image:



2. Training the Apriori Model on the dataset

To train the model, we will use the **apriori function** that will be imported from the **apyroi** package. This function will return the **rules** to train the model on the dataset. Consider the below code:

from apyori **import** apriori

rules= apriori(transactions= transactions, min\_support=0.003, min\_confidence = 0.2, min\_lift=3, min\_length=2, max\_length=2)

In the above code, the first line is to import the apriori function. In the second line, the apriori function returns the output as the rules. It takes the following parameters:

* **transactions**: A list of transactions.
* **min\_support**= To set the minimum support float value. Here we have used 0.003 that is calculated by taking 3 transactions per customer each week to the total number of transactions.
* **min\_confidence**: To set the minimum confidence value. Here we have taken 0.2. It can be changed as per the business problem.
* **min\_lift**= To set the minimum lift value.
* **min\_length**= It takes the minimum number of products for the association.
* **max\_length** = It takes the maximum number of products for the association.

### 3. Visualizing the result

Now we will visualize the output for our apriori model. Here we will follow some more steps, which are given below:

* **Displaying the result of the rules occurred from the apriori function**

results= list(rules)

results

As we can see, the above output is in the form that is not easily understandable. So, we will print all the rules in a suitable format.

* **Visualizing the rule, support, confidence, lift in more clear way:**

**for** item in results:

    pair = item[0]

    items = [x **for** x in pair]

    print("Rule: " + items[0] + " -> " + items[1])

    print("Support: " + str(item[1]))

    print("Confidence: " + str(item[2][0][2]))

    print("Lift: " + str(item[2][0][3]))

    print("=====================================")

By executing the above lines of code, we will get the below output:

**Rule: chicken -> light cream**

**Support: 0.004533333333333334**

**Confidence: 0.2905982905982906**

**Lift: 4.843304843304844**

**=====================================**

**Rule: escalope -> mushroom cream sauce**

**Support: 0.005733333333333333**

**Confidence: 0.30069930069930073**

**Lift: 3.7903273197390845**

**=====================================**

**Rule: escalope -> pasta**

**Support: 0.005866666666666667**

**Confidence: 0.37288135593220345**

**Lift: 4.700185158809287**

**=====================================**

**Rule: fromage blanc -> honey**

**Support: 0.0033333333333333335**

**Confidence: 0.2450980392156863**

**Lift: 5.178127589063795**

**=====================================**

**Rule: ground beef -> herb & pepper**

**Support: 0.016**

**Confidence: 0.3234501347708895**

**Lift: 3.2915549671393096**

**=====================================**

**Rule: tomato sauce -> ground beef**

**Support: 0.005333333333333333**

**Confidence: 0.37735849056603776**

**Lift: 3.840147461662528**

**=====================================**

**Rule: olive oil -> light cream**

**Support: 0.0032**

**Confidence: 0.20512820512820515**

**Lift: 3.120611639881417**

**=====================================**

**Rule: olive oil -> whole wheat pasta**

**Support: 0.008**

**Confidence: 0.2714932126696833**

**Lift: 4.130221288078346**

**=====================================**

**Rule: pasta -> shrimp**

**Support: 0.005066666666666666**

**Confidence: 0.3220338983050848**

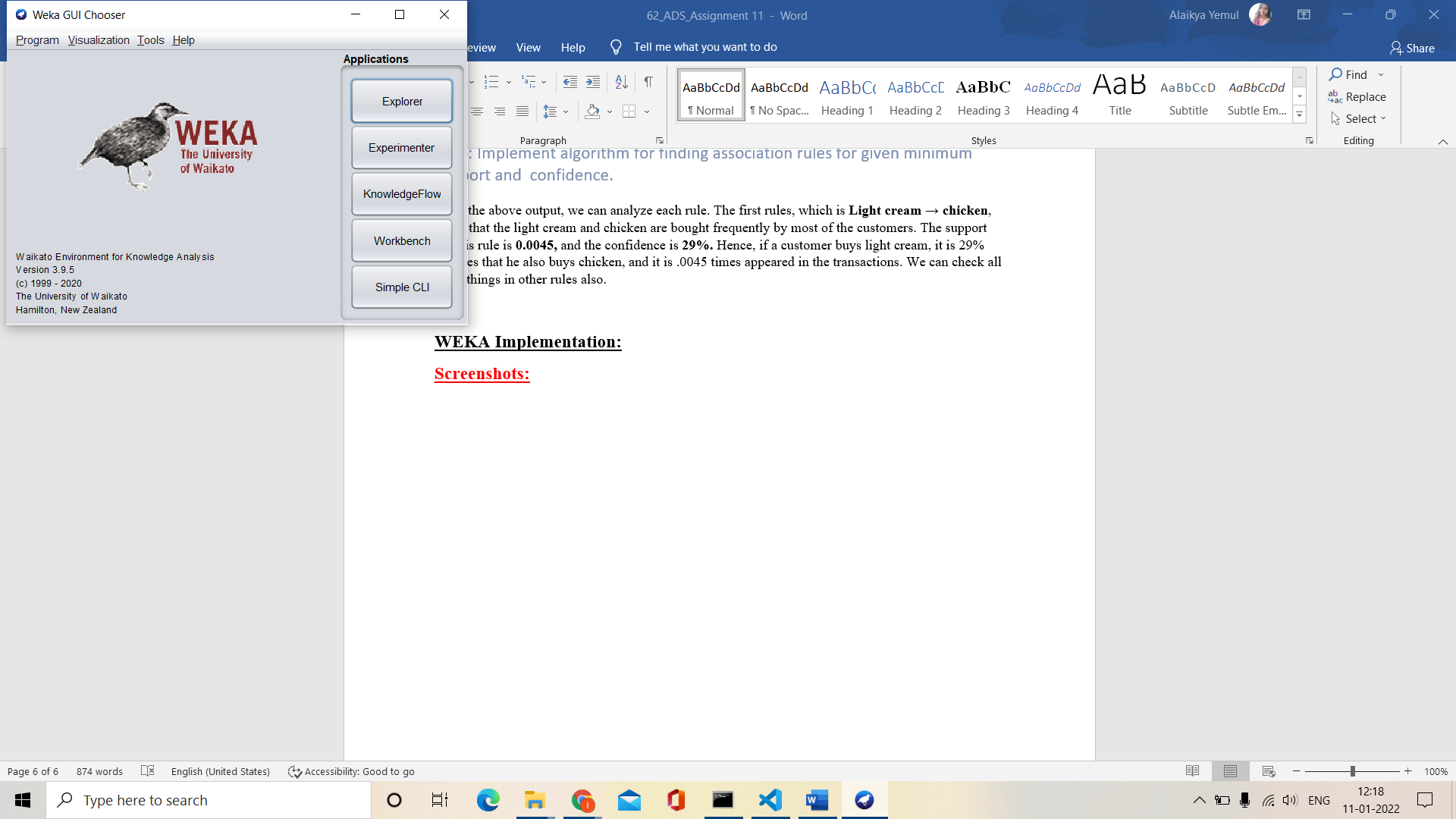
**Lift: 4.514493901473151**

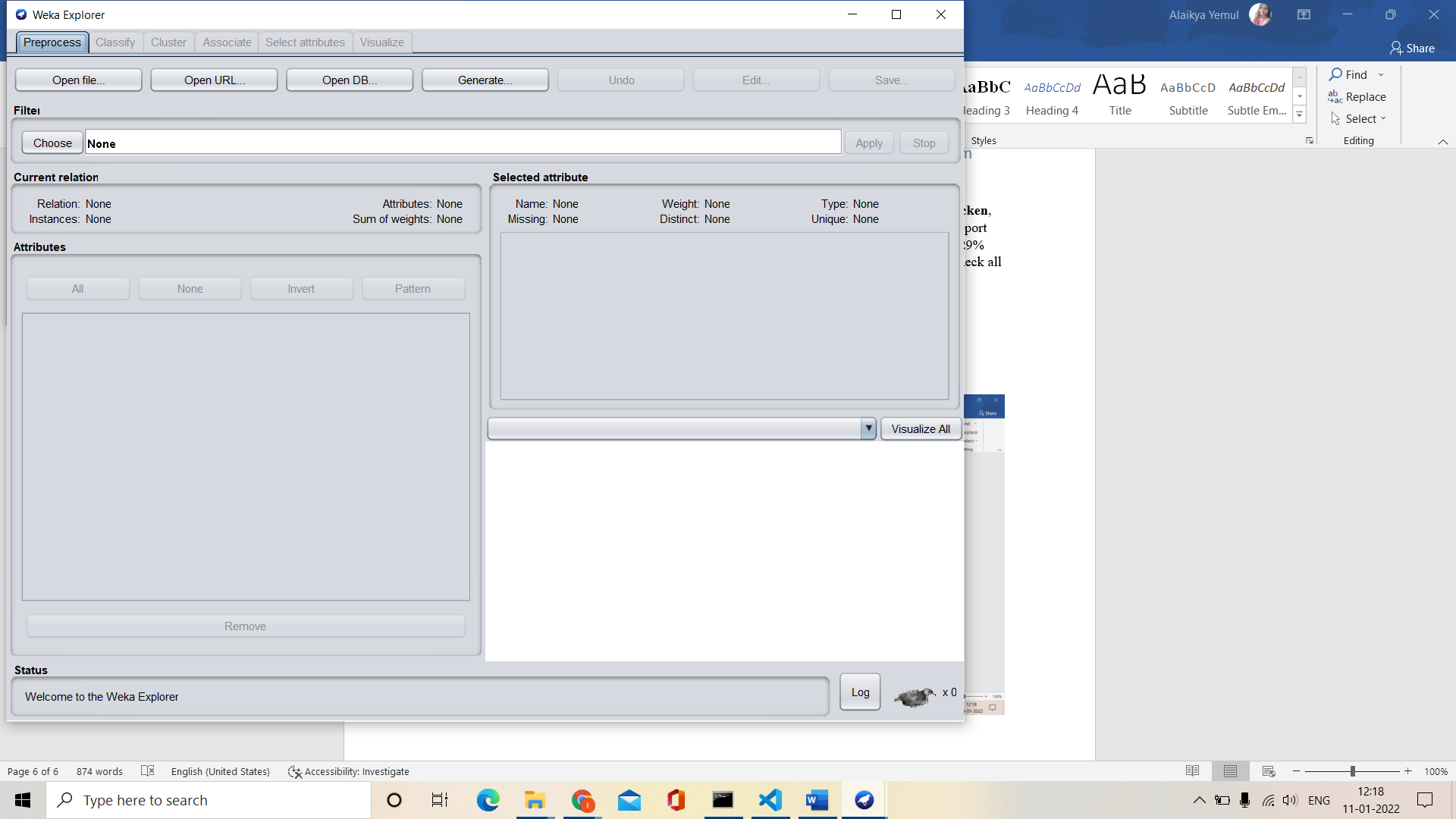
**=====================================**

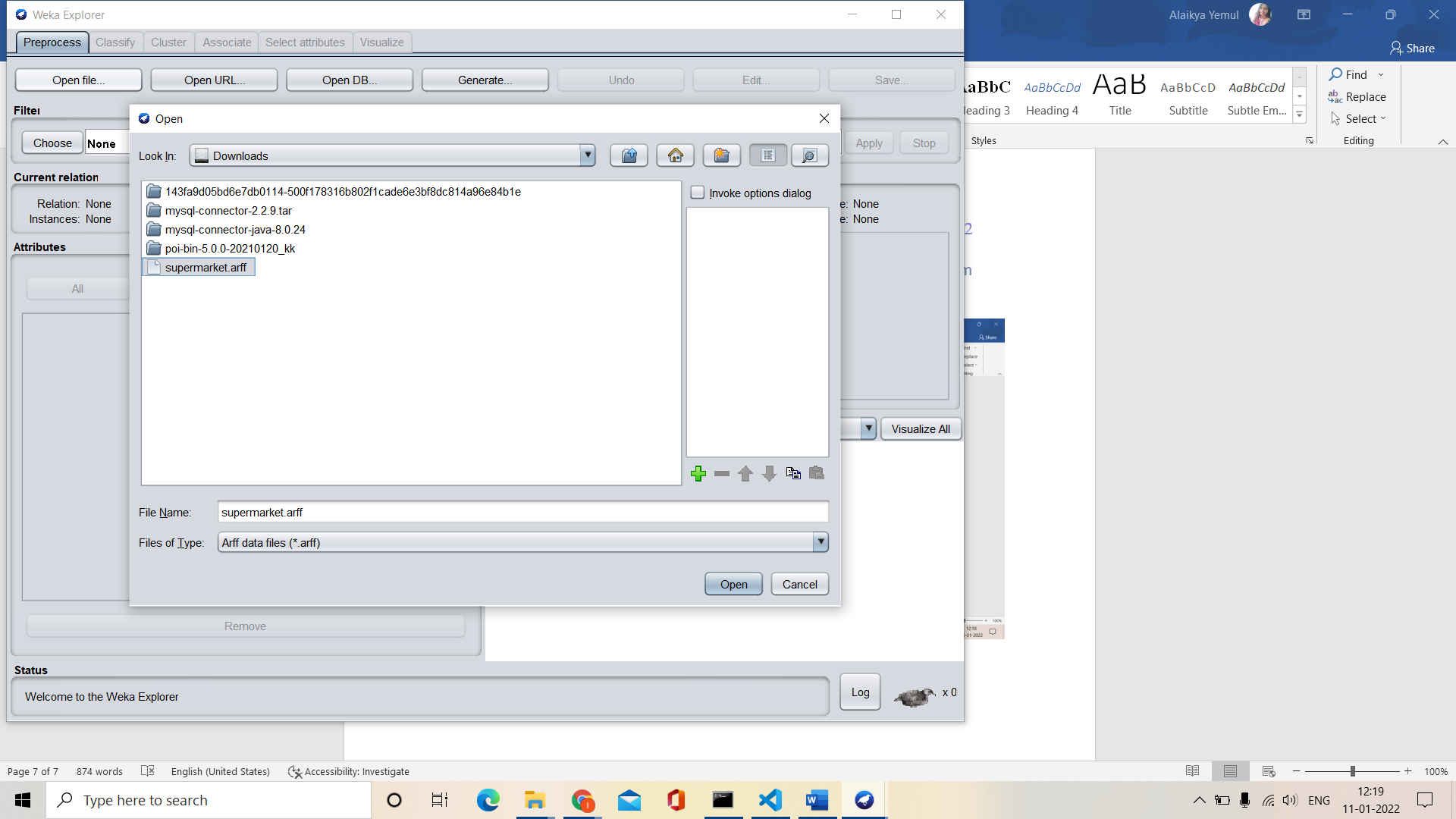
From the above output, we can analyze each rule. The first rules, which is **Light cream → chicken**, states that the light cream and chicken are bought frequently by most of the customers. The support for this rule is **0.0045,** and the confidence is **29%.** Hence, if a customer buys light cream, it is 29% chances that he also buys chicken, and it is .0045 times appeared in the transactions. We can check all these things in other rules also.

**WEKA Implementation:**

**Screenshots:**





****

