Data Exploration and Visualisation of Consumption patterns of households In Nepal

Data Mining, clustering and Segmentation approach .

by

Atit Bashyal

A report presented for the degree of
Masters of Science in Data Engineering

Prof. Dr. Adalbert F. X. Wilhelm
Jacobs University Bremen
Prof. Dr. Stefan Kettemann
Jacobs University Bremen

Contents

1 Executive Summary

1.1 Task Outline and Dataset

This project is conducted with two main objectives.

- The first objective of this project is to explore if Households in Nepal can be clustered meaningfully on the basis their average calorie consumption per month of various food-types.
- The second objective is to use an Association Mining approach (Market Basket Analysis) for each of the clusters to extract baskets where individual food items frequently bought together or grown together by households in each clusters identified.

The dataset used for this project is derived form the national-level data collected from Nepal National living Standards Survey(NLSS) 2011. The survey includes Household Level information from 3373 households sampled from different regions of Nepal. Information on the consumption of different food items by each sampled household is present in Section 5 (”Food Expenses and Household Production”) of the survey questionnaire. The survey questionnaire is included in Appendix A of this report.

1.2 Method of approach

With regards to the objective, data mining is performed on the raw dataset in order to create a comprehensive and tidy dataset. The first dataset, built for clustering,the average monthly consumption (in calories) of different food-types is computed. These food-types include : grains/cereals, pulses/lentils, poultry/dairy, oil/fat, vegetables, fruits, meat/fish and sugar.

After the clustering , the monthly average share of expenses on purchasing and expenses saved by using home-grown produce for each cluster is explored. For this purpose, data mining is again preformed in the raw dataset to extract information on individual household's monthly expenses(due to purchase for consumption)and saving(due to home-grown consumption) and cluster averages are computed .

Finally in order to find associations between different food items that are purchased or grown together, the dataset is transformed into the correct format so that it can be analyzed. For each clusters identified, the raw dataset is used to build 2 new sub-sets(for purchase and home production) where each row represents an household and each column represents a food-item.

The results from each objective discussed above are then subsequently displayed using meaningful visualisations. Ultimately, a summary of the key insights obtained from the data are discussed in the conclusion of this report.

2 Introduction

The purpose of this project is to explore consumption pattern of households in Nepal. The project is focused on exploration and primarily uses unsupervised learning technique to develop comprehensive insights from the available data. Data visualisations is used to convey these observations because graphical representations are an effective way to provide a clear understanding of the data to all humans due to the predominantly visual nature of our perceptual abilities. The software used for this project is Python. All the machine learning package used in the project are present in the open-source sklearn package available for python. Initially, this report will describe the existing data in order to familiarise readers with the available variables. Data description is important as a thorough understanding of the data is vital in understanding how the results have been produced. Subsequently, this report will address the data preprocessing methods used to create enriched tidy data-sets for analysis and subsequent visualisations. Data preprocessing plays a significant role in obtaining the calorie consumption different food groups. Furthermore, the report will then explore the process and choices of defining clusters of households based on their consumption of various food groups. Finally, some key observations are visualized and discussed.

3 The NLSS Dataset

The data for this project includes data and metadata from the Nepal Living Standard Survey(NLSS) conducted in the year 1995 .The NLSS survey is aimed at collecting data with the objective of measuring the living standards of the people in Nepal. The data is often utilized to determine the level of poverty in Nepal. The survey covers a wide range of topics related to household welfare which include demographic structure, monthly and weekly consumption, income, access to facilities, health, education etc. Results published from the NLSS survey has been of prime importance for government agencies and other organisations to assess the impact of policies and programs on socioeconomic changes in Nepal.

3.1 NLSS 1995 Data

The NLSS 1995 data obtained for this project included:

- The survey questionnaire
- Stata (version 14) data on household Food Expenses and Production

The Stata data file contains:

- The variable column "WWWHH" representing the household id
- The column "$S05A\_ITEM$" representing the 64 individual food items , the list of food items is available in Appendix A, of this report along with the survey questionnaire.
- Set of 10 response variables present in the survey data. Where each household has to answer consumption questions about individual food items. These questions represent each column of the original dataset. The questions are:

- $S05A\_02$: number of months Food item consumed by purchase
- $S05A\_03A$: monthly consumption amount of purchased food item
- $S05A\_03B$: consumption unit
- $S05A\_04$: amount spent in Purchase
- $S05A\_05$ number of months Food item consumed by home production
- $S05A\_06A$: monthly consumption amount of home grown food item
- $S05A\_06B$: consumption unit
- $S05A\_07$: amount typically would have spent if not home-grown
- $S05A\_08$: Value of food item received as gift/wage

	WWWHH	S05A_ITM	S05A_02	S05A_03A	S05A_03B	S05A_04	S05A_05	S05A_06A	S05A_06B	S05A_07	S05A_08
0	101.0	12	4.0	75.0	1.0	900.0	8.0	75.0	1.0	900.0	0.0
1	101.0	14	0.0	0.0	0.0	0.0	1.0	15.0	1.0	105.0	0.0
2	101.0	16	0.0	0.0	0.0	0.0	6.0	25.0	1.0	250.0	0.0
3	101.0	22	0.0	0.0	0.0	0.0	12.0	5.0	1.0	115.0	0.0
4	101.0	31	0.0	0.0	0.0	0.0	2.0	21.0	9.0	63.0	0.0

Figure 1: Snapshot of the original dataset with the first five rows shown

All of the variables are not relevant for answering the questions posed above. For building the monthly calorie consumption dataset for K-means The variables $S05A\_ITEM$, $S05A\_03A$, $S05A\_03B$, $S06A\_03A$ and$S05A_06B$ are used. Similarly to build the Association mining dataset, with monthly expenses, the variables $S05A\_ITEM$, $S05A\_04$ and $S05A\_07$ are used.

3.2 Exploration of Raw Data

An initial data exploration can be performed on the raw dataset in order to see the distribution of food items in terms of average monthly expense in purchasing it. For the home grown food items the cost reported to be possibly spent if not home-grown is taken as the proxy variable representing the monthly expense. These distributions of the Top 20 food items in terms of purchase and Home-growth can be seen in Figure 2 and 3 respectively.

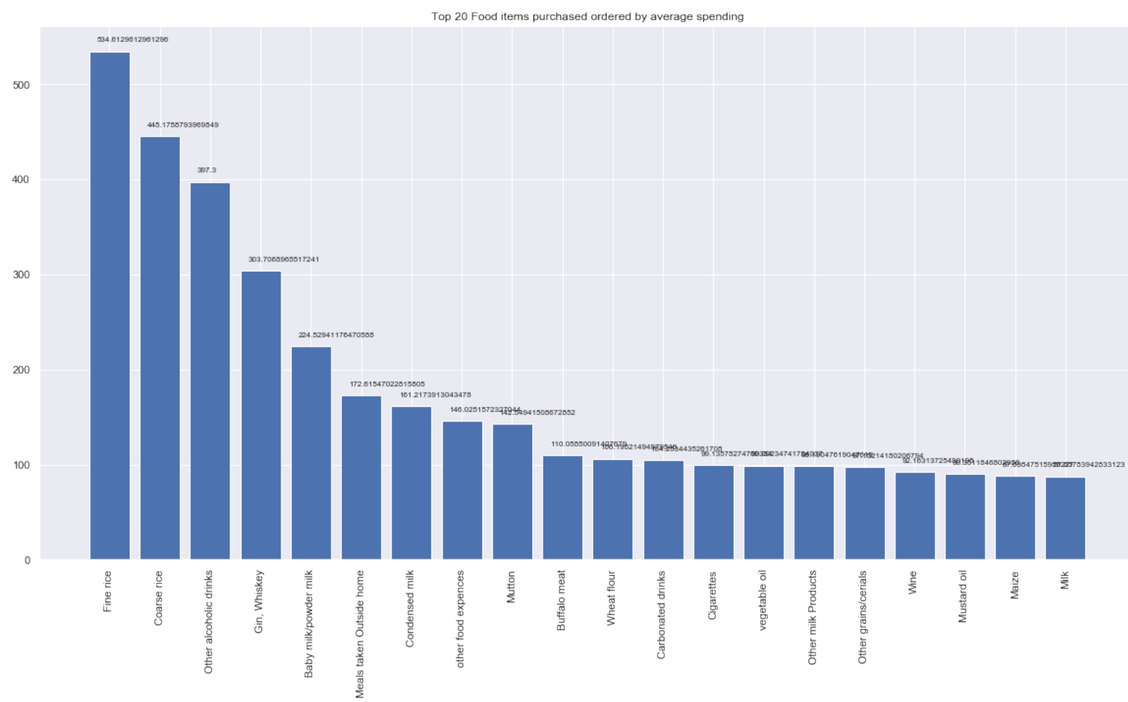

Figure 2: Bar-plot Top 20 food items purchased

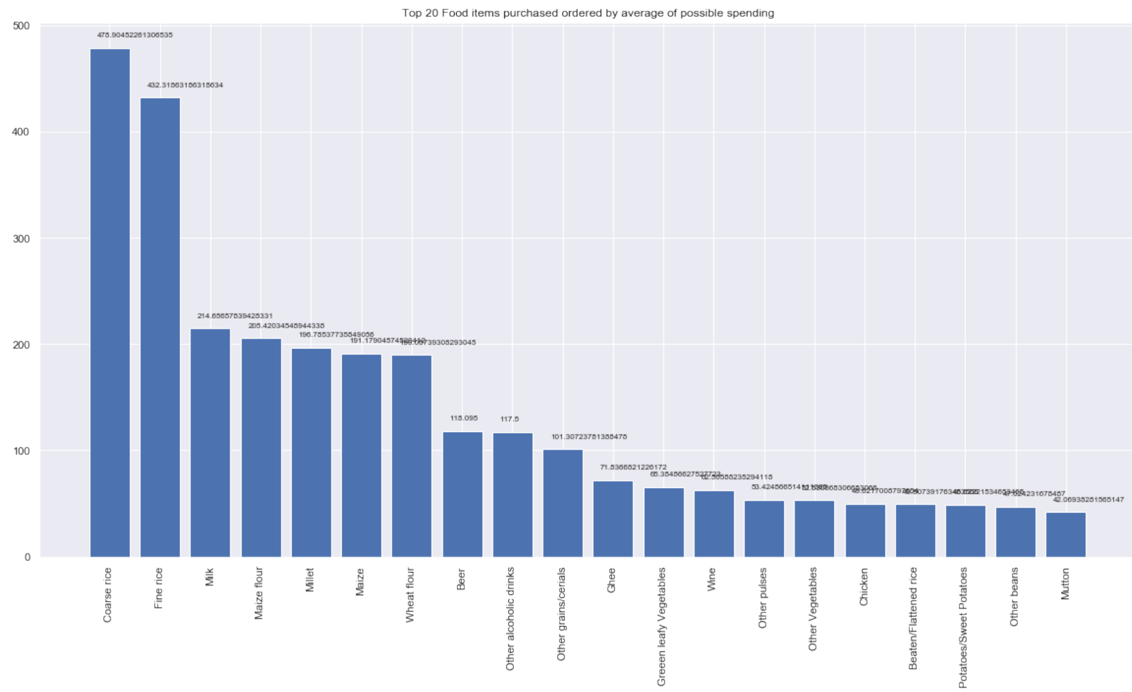

Figure 3: Bar-plot top 20 food items grown

4 Data Preprocessing

This section introduces and elaborates the process of cleaning and transforming the responses into a data format suitable for applying the learning algorithms. Since, the survey data used in this project was cleaned and stored in the Stata files, there were no problems to deal with regarding cleaning the data and addressing missing values. Therefore, the bulk of work done with regards to data preprocessing was to select the relevant variables for each objective , followed by the conversion of each column to its appropriate type.

4.1 Calorie Consumption dataset for K-means

As discussed earlier, the dataset built for clustering contains the average monthly consumption (in calories) of different food-types is computed. These food-types include : grains/cereals, pulses/lentils, poultry/dairy, oil/fat, vegetables, fruits, meat/fish and sugar. For each household to compute the average calorie intake of different food-types, consumption information on food items belonging to a particular food type are grouped together and averaged. Due to the discrepancies in the unit of food-consumed reported in the survey, all consumption values are first converted to grams per month. An overall average monthly calorie consumption per food-type is then computed by converting all monthly gram intake into Calories, using the gram to calorie conversion factor available for three macro-nutrients: Proteins, Carbohydrate and Fats. While computing the household average, information on whether the food item was purchased or home-grown is ignored. The First five rows of this dataset with household id set as the index can be seen in the figure below.

WWWHH	grains_cerials	pulses_lentals	dairy	oil_fat	vegetables	fruits	meat_fish	sugar
101.0	190000.0	5000.0	2100.0	500.0	37000.0	41200.0	1500.0	1000.0
104.0	315000.0	13000.0	15000.0	2000.0	93000.0	11300.0	6000.0	3000.0
105.0	148785.0	6547500.0	9220.0	2000.0	22000.0	1900.0	3000.0	2000.0
107.0	388109000.0	23987500.0	98100.0	5000.0	41000.0	19800.0	4000.0	5000.0
108.0	240000.0	5000.0	49050.0	1000.0	39000.0	10000.0	4000.0	0.0

Figure 4: Snapshot of the Calorie Consumption dataset dataset.

4.2 Market Basket Analysis (Association Mining)

Successively, the dataset must be transformed into the correct format so that it can be analyzed using an association mining approach. For this purpose the original dataset is first sliced using the purchase columns and home-production columns. Each dataset us then spread into a wider format with each row representing an household and each column representing a food item. The values of monthly consumption amount are then encoded into ones and zeroes denoting whether or not a item was consumed or not. Furthermore, households with reporting consumption of less than 5 items are excluded from the analysis at this step. The choice to exclude household with less than 5 items is taken as they are irrelevant to the analysis and the extremely high number of smaller food item sets will significantly affect the minimum support of the algorithm. The first five rows of the transformed data sets are shown in the figures below.

WWWHH	Fine rice	Coarse rice	Beaten/Flattened rice	Maize	Maize flour	Wheat flour	Millet	Other grains/cerials	Black pulse	Masoor	...	Apples	Pineapple	Papaya	Other fruits	Dried fruits	Fish
102.0	0	0.0	0.0	0.0	0	0.0	0	0	0.0	0.0	...	0.0	0.0	0	0	0	0.0
104.0	0	1.0	0.0	0.0	0	1.0	0	0	0.0	1.0	...	0.0	0.0	0	0	0	0.0
105.0	0	0.0	0.0	0.0	0	0.0	0	0	0.0	0.0	...	0.0	0.0	0	0	0	0.0
107.0	0	0.0	0.0	0.0	0	0.0	0	0	0.0	0.0	...	1.0	1.0	0	0	0	0.0
113.0	0	0.0	0.0	0.0	0	0.0	0	0	0.0	0.0	...	0.0	0.0	1	0	0	0.0

5 rows × 43 columns

Figure 5: Snapshot of the food item Purchase basket with the first five rows shown

	Fine rice	Coarse rice	Beaten/Flattened rice	Maize	Maize flour	Wheat flour	Millet	Other grains/cerials	Black pulse	Masoor	...	Apples	Pineapple	Papaya	Other fruits	Dried fruits	Fish
WWWHH																	
102.0	0	0.0	0.0	0.0	0	0.0	0	0	0.0	0.0	...	0.0	0.0	0	0	0	0.0
104.0	0	1.0	0.0	0.0	0	1.0	0	0	0.0	1.0	...	0.0	0.0	0	0	0	0.0
105.0	0	0.0	0.0	0.0	0	0.0	0	0	0.0	0.0	...	0.0	0.0	0	0	0	0.0
107.0	0	0.0	0.0	0.0	0	0.0	0	0	0.0	0.0	...	1.0	1.0	0	0	0	0.0
113.0	0	0.0	0.0	0.0	0	0.0	0	0	0.0	0.0	...	0.0	0.0	1	0	0	0.0

5 rows × 43 columns

Figure 6: Snapshot of the food item home-production basket with the first five rows shown

5 Analysis and Exploration

5.1 K-means clustering based on Calorie Consumption

To start with the clustering process, the optimal number of clusters to be made is decided using the elbow method. The sum of the squared differences between the observations and the corresponding centroids (Distortion) is plotted in the Y-axis against number of clusters k in the x-axis.

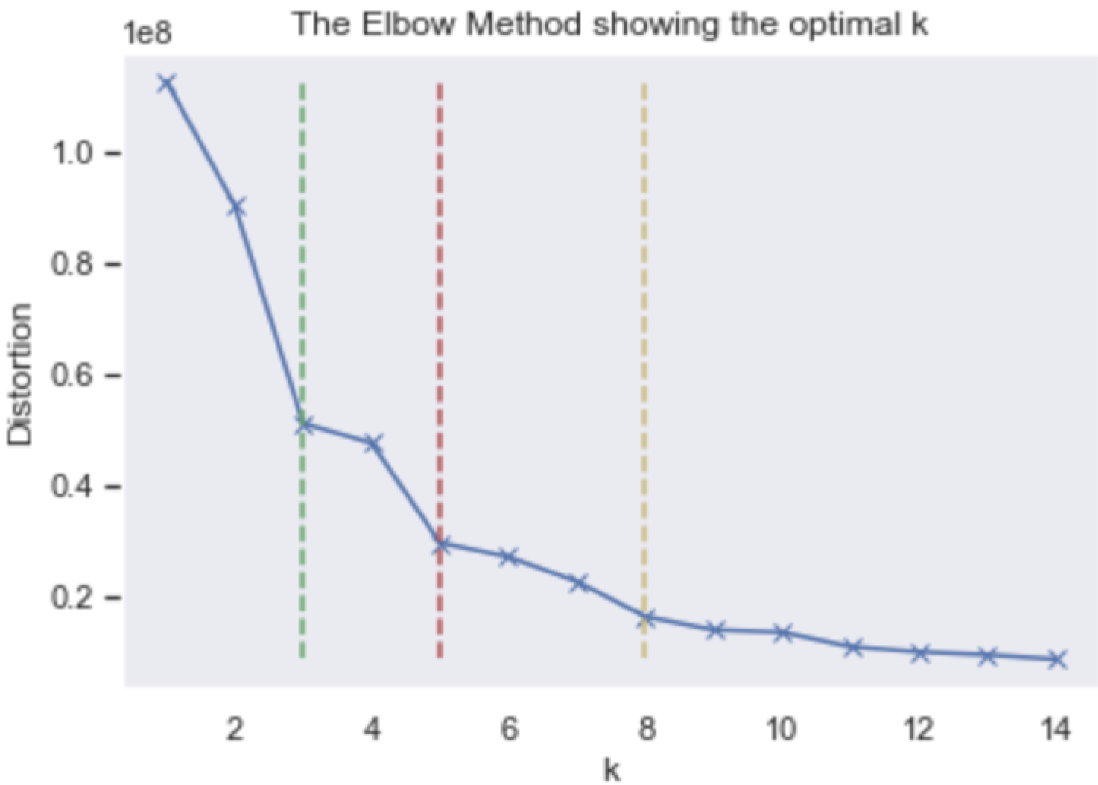

Figure 7: Elbow plot, for optimal k

According to the elbow method plot, the significant bends at k=3, 5 and 8 show that the loss function has local minima at these points. Using cluster numbers at these points is expected to produce meaningful clusters but Experimenting with the k value show that using k = 8 resulted in the classes not being well separated with extremely overlapping classes k=3 resulted in high cluster imbalance. In the case of k=8, The increased flexibility of the clusters was over-fitting our data. Likewise, using k=3 allowed restricted the algorithm to discriminate between sub-clusters. Therefore, five clusters have been chosen for this analysis. The cluster grouping is used to compute the average share of each food-type in the total calorie consumption. The values of the average share of each food-type for each cluster are shown in the figure below.

cluster_label	grains_cerials	pulses_lentals	poultry_dairy	oil_fat	vegetables	fruits	meat_fish	sugar
2	86.377041	0.727356	0.019088	0.215470	12.653670	0.003675	0.002659	0.001040
1	66.076251	1.660805	0.062673	0.001497	32.191960	0.004629	0.001586	0.000600
3	59.300297	0.971895	38.841942	0.345977	0.528823	0.007749	0.002294	0.001023
0	73.814927	2.965131	6.920791	1.220846	8.119519	4.203408	1.541554	1.213825
4	54.475372	4.201936	6.198237	0.832427	8.597867	22.861464	0.895126	1.937572

Figure 8: Average share of food-types for Clusters Identified

All the clusters exhibit a 'traditional-developing-country' diet, with grain and cereals having the maximum weight on the total average calorie intake. But there is an interesting tend that can be seen across the clusters:

- Clusters 2, 1, and 3 are clusters where grains and cereals are the primary food group to fulfill calorie consumption. Each of these clusters have secondary food group with a significant contribution in the calorie consumption, while other food groups make minor/ignore-able contribution.
- Cluster 0 is also the cluster where grain has a major contribution in the calorie intake, but shows a trend where all other food groups have significant contributions to fulfill the calorie intake. It shows a non-traditional behaviour of diet pattern in which animal products and other foods play a significant role in diet.
- Cluster 4 shows the most interesting trend. In this cluster has the lowest reliance on grain and cereals of all the clusters. Fruits have the second most contribution in the calorie intake, with Meat/Fish and Fat having the least contribution in calorie intake. These households have Non-traditional diet pattern as and also show a trend seen 'modern-diet' where people consume the least amount of Fats and meat products.

The figure below shows the average share of monetary value on Purchase and saving from home-production.

Figure 9: share of monetary value on Purchase and saving from home production

Looking at the figure it is evident that clusters 1, 2, 3 and 4 are clusters where higher share of the food items consumed are home-grown products. While cluster 0 is the only cluster where higher share of food items are consumed by purchase.

Based on these evidences the final grouping of the clusters are as follows: Cluster 1, 2 and 3 are grouped together as Traditional-Home-Producers category, Cluster 4 is taken as the Non-traditional-Home-Producer category and finally, cluster 0 is the Non-traditional-Purchaser category.

5.2 Exploration of average share of cost on home grown and Purchased food-types

After the clustering , the average share of cost on home grown and Purchased food-types is explored for each clusters. For this purpose, data mining is again preformed in the raw dataset to extract information on individual household's monthly cost incurred (due to purchase for consumption)and cost saving(due to home-grown consumption) on the different food-types, grouped together by clusters and averaged.This exploration aids the goal of this project in two ways:

- First, it helps us understand how the total consumption cost is distributed across different food-types in for each cluster.
- Second, it helps us gain an overview on food-types are that are consumed by purchase and growth by each cluster before Association Mining on individual food items is preformed

5.2.1 Traditional-Home-Producers

As discussed above, This cluster includes households with higher share of the food items consumed by home-production. It is evident from the figure that, the share of cost for grain and cereals in both purchase group and Home-production group is almost equal. It is interesting to see that this consumption of Poultry/Dairy and Vegetables for this clusters are fulfilled mostly by home

Production while consumption of fat/oil, meat products and sugar are fulfilled by purchase. The remaining food types have almost similar share in home production and purchase. The figure below compares the average share of cost on home grown and Purchased food-types for this cluster.

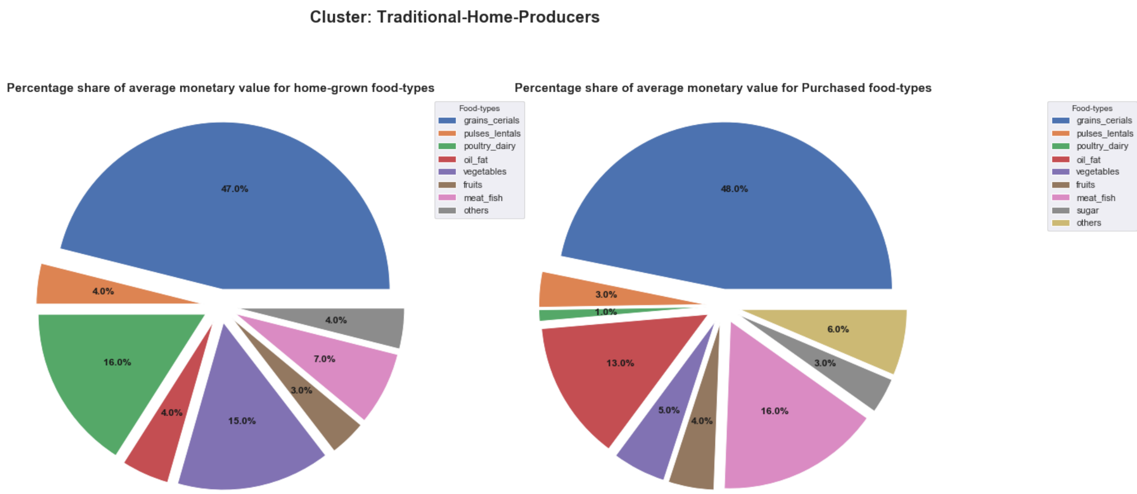

Figure 10

5.2.2 Non-traditional-Home-Producer

This cluster also includes households where higher share of the food items consumed are home-grown products. This cluster was more interesting in terms of the share that each food items had in terms of calorie consumption. Compared to the traditional cluster this cluster represented an modern-diet trend with less reliance on grains/cereals and significant contributions from other food groups excluding Meat/Fish and Fat. It is seen that grains and cereals and fruits are the food group that are consumed mostly by home-production in this cluster. Also meat/fish, sugar and oil/fat are the food group that are mostly consumed by purchase. Other food groups have almost similar distributions in consumption by purchase and home-production. The figure below compares the average share of cost on home grown and Purchased food-types for this cluster.

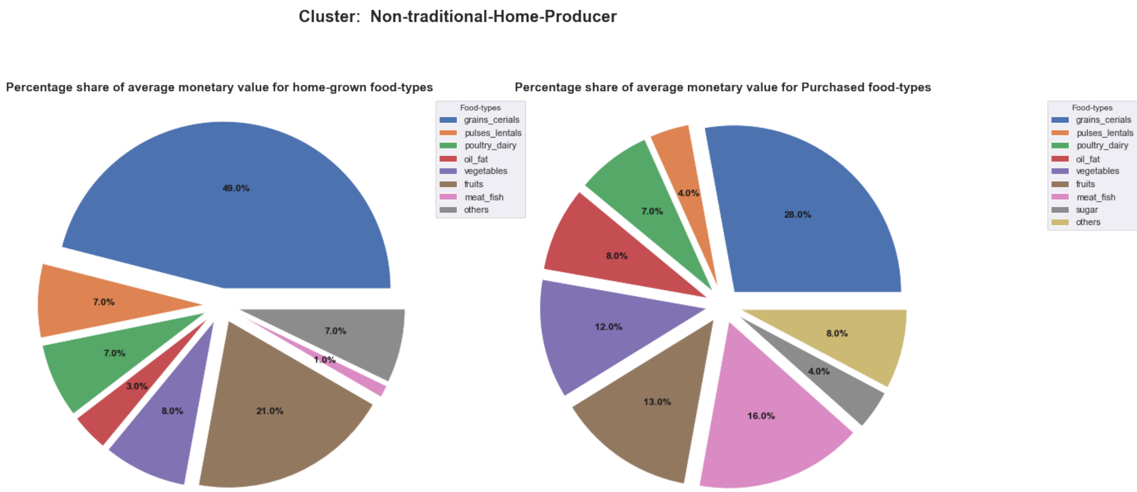

Figure 11

5.2.3 Non-traditional-Purchaser

This cluster includes households where higher share of the food items consumed are purchased products. In terms of consumption in this cluster grain and cereals still have a major contribution in the calorie intake, but compared to the traditional cluster all other food groups also have significant contributions to fulfill the calorie intake. It shows a non-traditional behaviour of diet pattern in which animal products and other foods play a significant role in diet.It is still seen that grains and cereals are the food group that are consumed mostly by home-production in this cluster. It is interesting to see that unlike in other clusters, this cluster has significant share from different food groups in consumption in both purchase and home-production. It is also interesting to see that even in this cluster, meat/fish, sugar and oil/fat are the food group that are mostly consumed by purchase rather than home purchase. The figure below compares the average share of cost on home grown and Purchased food-types for this cluster.

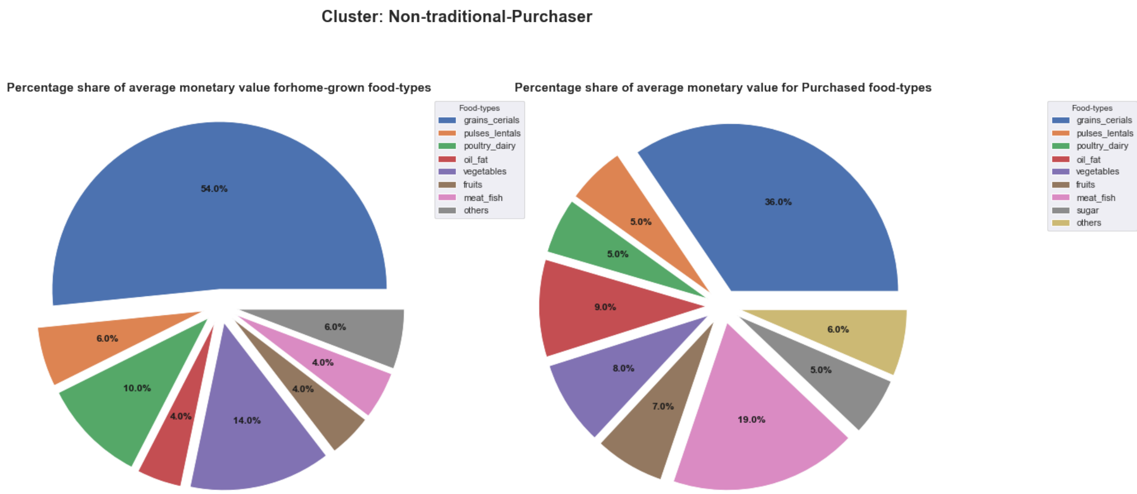

Figure 12

It is interesting to notice that across the three clusters, the food groups that has the highest share in calorie consumption are also the food groups that are mostly consumed by home-growth. Similarly, meat/Fish and Oil/Fat in each of the clusters are consumed in higher amount by purchasing the food items. It is also seen that Sugar is the food item that is consumed only by purchase. Pulses/lentils and dairy/poultry are consumed in higher amount by home growth than by purchase.

5.3 Association Mining: Purchased and Home Grown Food baskets by clusters

After the data has been transformed into the correct format for each clusters as discussed above, association mining is performed using the Apriori algorithm. The Apriori algorithm identifies relationships between frequent individual food items in the entire itemset by observing the frequency in which these subset of items are consumed . This method was chosen for our market basket analysis as it is devised to work well with datasets with a large number of items. The datasets are then fit to an Apriori algorithm with a minimum support of 0.5. The support is defined for the itemset and measures the frequency that an item occurs in a dataset. It is defined by the following formula :

$$supp(X) = \frac{CountofTransactionsIncludingX}{TotalTransactions} \quad (1)$$

The association rules are then determined by a confidence metric with a minimum threshold of 0.5. The confidence metric measures the probability of observing the consequent Y in an order, given that the order also contains the antecedent X. It is defined using the following formula :

$$conf(X \rightarrow Y) = \frac{supp(XY)}{supp(X)} \quad (2)$$

Another metric that can be used to determine whether or not rules can be derived is lift, which takes into account the popularity of both item sets. It is defined as :

$$lift(X \rightarrow Y) = \frac{supp(XY)}{supp(X) * supp(Y)} \tag{3}$$

If the lift is greater than 1, it means that item set Y is likely to be bought with item set X. If lift is less than 1, it means that the presence of item set X could hurt the chances of item set Y being bought. In the analysis all associations with a lift of less than 1 have been excluded. The resulting associations shown from of the basket analysis for each clusters are shown in the in the sub sections below.

5.3.1 Traditional-Home-Producers

	antecedents	consequents	antecedent support	consequent support	support	confidence	lift	leverage
13023	(Potatoes/Sweet Potatoes, Onions, Apples)	(Citrus Fruits, Cauliflower/Cabbage)	0.569959	0.728395	0.500000	0.877256	1.204369	0.084845
6906	(Potatoes/Sweet Potatoes, Apples)	(Mangoes, Citrus Fruits)	0.592593	0.710905	0.506173	0.854167	1.201520	0.084896
6778	(Potatoes/Sweet Potatoes, Apples)	(Citrus Fruits, Tomatoes)	0.592593	0.701646	0.502058	0.847222	1.207478	0.086267
13033	(Potatoes/Sweet Potatoes, Apples)	(Citrus Fruits, Onions, Cauliflower/Cabbage)	0.592593	0.695473	0.500000	0.843750	1.213203	0.087868
7201	(Onions, Apples)	(Citrus Fruits, Tomatoes)	0.593621	0.701646	0.500000	0.842288	1.200445	0.083488
13037	(Onions, Apples)	(Potatoes/Sweet Potatoes, Citrus Fruits, Cauli...	0.593621	0.694444	0.500000	0.842288	1.212894	0.087763
17295	(Bananas, Milk, Cauliflower/Cabbage)	(Mangoes, Potatoes/Sweet Potatoes, Tomatoes, O...	0.613169	0.687243	0.508230	0.828859	1.206064	0.086835
7205	(Apples)	(Citrus Fruits, Tomatoes, Onions)	0.627572	0.663580	0.500000	0.796721	1.200640	0.083556
7192	(Citrus Fruits, Tomatoes, Onions)	(Apples)	0.663580	0.627572	0.500000	0.753488	1.200640	0.083556
17234	(Mangoes, Potatoes/Sweet Potatoes, Tomatoes, O...	(Bananas, Milk, Cauliflower/Cabbage)	0.687243	0.613169	0.508230	0.739521	1.206064	0.086835
13028	(Potatoes/Sweet Potatoes, Citrus Fruits, Cauli...	(Onions, Apples)	0.694444	0.593621	0.500000	0.720000	1.212894	0.087763
13032	(Citrus Fruits, Onions, Cauliflower/Cabbage)	(Potatoes/Sweet Potatoes, Apples)	0.695473	0.592593	0.500000	0.718935	1.213203	0.087868
6779	(Citrus Fruits, Tomatoes)	(Potatoes/Sweet Potatoes, Apples)	0.701646	0.592593	0.502058	0.715543	1.207478	0.086267
7196	(Citrus Fruits, Tomatoes)	(Onions, Apples)	0.701646	0.593621	0.500000	0.712610	1.200445	0.083488
6903	(Mangoes, Citrus Fruits)	(Potatoes/Sweet Potatoes, Apples)	0.710905	0.592593	0.506173	0.712012	1.201520	0.084896
13042	(Citrus Fruits, Cauliflower/Cabbage)	(Potatoes/Sweet Potatoes, Onions, Apples)	0.728395	0.569959	0.500000	0.686441	1.204369	0.084845

Figure 13: Association Basket Purchased items : Traditional-Home Producers

	antecedents	consequents	antecedent support	consequent support	support	confidence	lift	leverage
12273	(Maize, Ghee, Cauliflower/Cabbage)	(Tomatoes, Milk)	0.540816	0.806122	0.510204	0.943396	1.170289	0.074240
19784	(Maize, Potatoes/Sweet Potatoes, Ghee, Caulifl...	(Tomatoes, Milk)	0.530612	0.806122	0.500000	0.942308	1.168939	0.072262
15731	(Bananas, Tomatoes, Onions)	(Potatoes/Sweet Potatoes, Cauliflower/Cabbage)	0.591837	0.724490	0.500000	0.844828	1.166100	0.071220
22741	(Bananas, Ghee, Tomatoes)	(Maize, Potatoes/Sweet Potatoes, Milk, Mustard...	0.622449	0.714286	0.520408	0.836066	1.170492	0.075802
7272	(Cauliflower/Cabbage, Mustard oil)	(Tomatoes, Onions)	0.642857	0.693878	0.530612	0.825397	1.189542	0.084548
15637	(Mustard oil, Potatoes/Sweet Potatoes, Caulifl...	(Tomatoes, Onions)	0.632653	0.693878	0.520408	0.822581	1.185484	0.081424
5116	(Maize, Cauliflower/Cabbage)	(Tomatoes, Onions)	0.653061	0.693878	0.530612	0.812500	1.170956	0.077468
13114	(Maize, Potatoes/Sweet Potatoes, Cauliflower/C...	(Tomatoes, Onions)	0.642857	0.693878	0.520408	0.809524	1.166667	0.074344
15651	(Mustard oil, Cauliflower/Cabbage)	(Potatoes/Sweet Potatoes, Tomatoes, Onions)	0.642857	0.683673	0.520408	0.809524	1.184080	0.080904
5117	(Tomatoes, Onions)	(Maize, Cauliflower/Cabbage)	0.693878	0.653061	0.530612	0.764706	1.170956	0.077468
7273	(Tomatoes, Onions)	(Cauliflower/Cabbage, Mustard oil)	0.693878	0.642857	0.530612	0.764706	1.189542	0.084548
15634	(Potatoes/Sweet Potatoes, Tomatoes, Onions)	(Mustard oil, Cauliflower/Cabbage)	0.683673	0.642857	0.520408	0.761194	1.184080	0.080904
13131	(Tomatoes, Onions)	(Maize, Potatoes/Sweet Potatoes, Cauliflower/C...	0.693878	0.642857	0.520408	0.750000	1.166667	0.074344
15648	(Tomatoes, Onions)	(Mustard oil, Potatoes/Sweet Potatoes, Caulifl...	0.693878	0.632653	0.520408	0.750000	1.185484	0.081424
22688	(Maize, Potatoes/Sweet Potatoes, Milk, Mustard...	(Bananas, Ghee, Tomatoes)	0.714286	0.622449	0.520408	0.728571	1.170492	0.075802
15734	(Potatoes/Sweet Potatoes, Cauliflower/Cabbage)	(Bananas, Tomatoes, Onions)	0.724490	0.591837	0.500000	0.690141	1.166100	0.071220
12292	(Tomatoes, Milk)	(Maize, Ghee, Cauliflower/Cabbage)	0.806122	0.540816	0.510204	0.632911	1.170289	0.074240
19833	(Tomatoes, Milk)	(Maize, Potatoes/Sweet Potatoes, Ghee, Caulifl...	0.806122	0.530612	0.500000	0.620253	1.168939	0.072262

Figure 14: Association Basket Home-grown items: Traditional-Home-Producers

5.3.2 Non-traditional-Home-Producer

	antecedents	consequents	antecedent support	consequent support	support	confidence	lift	leverage
16	(Wheat flour)	(Mustard oil)	0.529412	0.823529	0.529412	1.000000	1.214286	0.093426
10314	(Mangoes, Tomatoes, Milk, Apples)	(Citrus Fruits)	0.529412	0.823529	0.529412	1.000000	1.214286	0.093426
10137	(Mangoes, Citrus Fruits, Tomatoes)	(Bananas, Milk)	0.588235	0.823529	0.588235	1.000000	1.214286	0.103806
4229	(Curd, Potatoes/Sweet Potatoes, Onions)	(Citrus Fruits)	0.529412	0.823529	0.529412	1.000000	1.214286	0.093426
5602	(Potatoes/Sweet Potatoes, Apples, Mutton)	(Citrus Fruits)	0.588235	0.823529	0.588235	1.000000	1.214286	0.103806
...	...	...	...	...	...	...	...	...
1831	(Mustard oil)	(Beaten/Flattened rice, Milk, Mutton)	0.823529	0.529412	0.529412	0.642857	1.214286	0.093426
11989	(Citrus Fruits)	(Bananas, Tomatoes, Apples, Mutton)	0.823529	0.529412	0.529412	0.642857	1.214286	0.093426
1839	(Milk, Mustard oil)	(Wheat flour, Mutton)	0.823529	0.529412	0.529412	0.642857	1.214286	0.093426
11929	(Citrus Fruits)	(Mangoes, Bananas, Tomatoes, Apples)	0.823529	0.529412	0.529412	0.642857	1.214286	0.093426
14623	(Citrus Fruits)	(Potatoes/Sweet Potatoes, Mutton, Cauliflower/...	0.823529	0.529412	0.529412	0.642857	1.214286	0.093426

4716 rows × 8 columns

Figure 15: Association Basket Purchased items: Non-Traditional-Home-Producers

	antecedents	consequents	antecedent support	consequent support	support	confidence	lift	leverage
38	(Coarse rice, Milk)	(Wheat flour)	0.500000	0.888889	0.500000	1.0000	1.125000	0.055556
62	(Potatoes/Sweet Potatoes, Milk)	(Wheat flour)	0.583333	0.888889	0.583333	1.0000	1.125000	0.064815
17	(Milk)	(Wheat flour)	0.722222	0.888889	0.722222	1.0000	1.125000	0.080247
15	(Rahar)	(Wheat flour)	0.500000	0.888889	0.500000	1.0000	1.125000	0.055556
67	(Mangoes, Milk)	(Wheat flour)	0.583333	0.888889	0.583333	1.0000	1.125000	0.064815
...	...	...	...	...	...	...	...	...
46	(Wheat flour)	(Mangoes, Coarse rice)	0.888889	0.611111	0.555556	0.6250	1.022727	0.012346
18	(Wheat flour)	(Mustard oil)	0.888889	0.555556	0.500000	0.5625	1.012500	0.006173
14	(Wheat flour)	(Rahar)	0.888889	0.500000	0.500000	0.5625	1.125000	0.055556
11	(Wheat flour)	(Beaten/Flattened rice)	0.888889	0.527778	0.500000	0.5625	1.065789	0.030864
39	(Wheat flour)	(Coarse rice, Milk)	0.888889	0.500000	0.500000	0.5625	1.125000	0.055556

70 rows × 8 columns

Figure 16: Association Basket Home-grown items: Non-Traditional-Home-Producers

5.3.3 Non-traditional-Purchaser

	antecedents	consequents	antecedent support	consequent support	support	confidence	lift	leverage
15827	(Onions, Apples)	(Bananas, Potatoes/Sweet Potatoes, Citrus Fruits)	0.608324	0.688367	0.506937	0.833333	1.210594	0.088186
15823	(Potatoes/Sweet Potatoes, Apples)	(Bananas, Citrus Fruits, Onions)	0.610459	0.686233	0.506937	0.830420	1.210114	0.088020
15943	(Potatoes/Sweet Potatoes, Apples)	(Mangoes, Citrus Fruits, Onions)	0.610459	0.684098	0.502668	0.823427	1.203667	0.085054
15707	(Onions, Apples)	(Potatoes/Sweet Potatoes, Citrus Fruits, Tomat...	0.608324	0.685165	0.500534	0.822807	1.200888	0.083731
15703	(Potatoes/Sweet Potatoes, Apples)	(Citrus Fruits, Tomatoes, Onions)	0.610459	0.677695	0.500534	0.819930	1.209881	0.086829
15822	(Bananas, Citrus Fruits, Onions)	(Potatoes/Sweet Potatoes, Apples)	0.686233	0.610459	0.506937	0.738725	1.210114	0.088020
15702	(Citrus Fruits, Tomatoes, Onions)	(Potatoes/Sweet Potatoes, Apples)	0.677695	0.610459	0.500534	0.738583	1.209881	0.086829
15818	(Bananas, Potatoes/Sweet Potatoes, Citrus Fruits)	(Onions, Apples)	0.688367	0.608324	0.506937	0.736434	1.210594	0.088186
15942	(Mangoes, Citrus Fruits, Onions)	(Potatoes/Sweet Potatoes, Apples)	0.684098	0.610459	0.502668	0.734789	1.203667	0.085054
15698	(Potatoes/Sweet Potatoes, Citrus Fruits, Tomat...	(Onions, Apples)	0.685165	0.608324	0.500534	0.730530	1.200888	0.083731

Figure 17: Association Basket Purchased items: Non-traditional-Purchaser

	antecedents	consequents	antecedent support	consequent support	support	confidence	lift	leverage
24385	(Potatoes/Sweet Potatoes, Maize, Ghee, Wheat f...	(Milk)	0.5625	0.9250	0.5625	1.000000	1.081081	0.042187
13471	(Maize, Wheat flour, Ghee, Tomatoes)	(Milk)	0.6125	0.9250	0.6125	1.000000	1.081081	0.045937
13531	(Maize, Wheat flour, Ghee, Papaya)	(Milk)	0.5125	0.9250	0.5125	1.000000	1.081081	0.038437
25872	(Potatoes/Sweet Potatoes, Maize, Onions, Ghee,...	(Milk)	0.5000	0.9250	0.5000	1.000000	1.081081	0.037500
25828	(Maize, Bananas, Ghee, Mustard oil)	(Potatoes/Sweet Potatoes, Milk)	0.5500	0.9125	0.5500	1.000000	1.095890	0.048125
...	...	...	...	...	...	...	...	...
30029	(Potatoes/Sweet Potatoes)	(Maize, Ghee, Milk, Wheat flour, Bananas, Must...	0.9875	0.5000	0.5000	0.506329	1.012658	0.006250
20575	(Potatoes/Sweet Potatoes)	(Mustard oil, Bananas, Ghee, Cauliflower/Cabbage)	0.9875	0.5000	0.5000	0.506329	1.012658	0.006250
9223	(Potatoes/Sweet Potatoes)	(Bananas, Tomatoes, Papaya)	0.9875	0.5000	0.5000	0.506329	1.012658	0.006250
2190	(Potatoes/Sweet Potatoes)	(Mangoes, Onions)	0.9875	0.5000	0.5000	0.506329	1.012658	0.006250
3274	(Potatoes/Sweet Potatoes)	(Mangoes, Wheat flour, Coarse rice)	0.9875	0.5000	0.5000	0.506329	1.012658	0.006250

18167 rows × 8 columns

Figure 18: Association Basket Home-grown items: Non-traditional-Purchaser

6 Summary and conclusion

The primary objective of this project was to explore if Households in Nepal can be clustered meaningfully on the basis their average calorie consumption per month of various food-types. The results of this exploration show that the households can be clustered into two primary groups: Home-producers and Purchasers with regards to the share of cost on purchase and home production. The Home-produces can further be clustered into two groups: Traditional and Non-Traditional based on the share of calories consumed according to different food types. The explorations also show that irrespective of the clusters, Grains and Cereals still have the highest share in fulfilling the calorie intake of households in Nepal. It is also seen that across clusters, the food groups that have the highest share in calorie consumption are also the food groups that are mostly consumed by home-growth. Similarly, meat/Fish and Oil/Fat in each of the clusters are consumed in higher amount by purchasing. It is also seen that Sugar is the food item that is consumed only by purchase. Pulses/lentils and dairy/poultry are consumed in higher amount by home growth than by purchase.

References

[1] J. K. 'Andrea Carlson' and C. Nadav', *Consumers' Retail Source of Food: A Cluster Analysis.* 2018-09-1.

[2] M. F. C. R. J. F. t. H. T. O. 'KarinFAMHuishof', 'MWedel', *Clustering of dietary variables and other lifestyle factors (Dutch Nutritional Surveillance System).*

[3] H. Jaeger, *Principles of Statistical Modelling lecture notes spring 2019.*

[4] A. N. Rae, *Food consumption patterns and nutrition in urban Java households: the discriminatory power of some socioeconomic variables.*

[5] J. F. "Trevor hastie", "Robert Tibshirani", *The Elements of Statistical Learning: Data Mining, Inference and Prediction.*

[6] J. K. 'NIKOLAOS KATSARAS', 'PAUL WOLFSON' and B. SENAUER', *DATA MINING A SEGMENTATION ANALYSIS OF U.S. GROCERY SHOPPERS.*

[7] B. of Statistics, *National Living Standards Survey 1995, Questionnaire.*

A NLSS QUESTIONNAIRE

SECTION 5

FOOD EXPENSES AND HOME PRODUCTION

1. Have you consumed ..[FOOD].. during the past 12 months? PUT A CHECK (✓) IN THE APPROPRIATE BOX FOR EACH FOOD ITEM. IF THE ANSWER TO Q. 1 IS YES, ASK Q. 2-8.				FOOD PURCHASES: 2. How many months in the past 12 months did you purchase ..[FOOD].. ? IF NONE WRITE ZERO AND ➔5	3. In a typical month during which you purchased ..[FOOD]. how much did you purchase?		4. How much would you normally have to spend in total to buy this quantity?	HOME PRODUCTION: 5. How many months in the past 12 months did you consume ..[FOOD].. that you grew or produced yourself? IF NONE WRITE ZERO AND ➔8	6. In a typical month during which you ate ..[FOOD].., how much did your household consume of ..[FOOD]..?		7. How much would your household have to spend in the market to buy this quantity of ..[FOOD].. (i.e. the amount consumed in a typical month)?	IN-KIND: 8. What is the total value of the ..[FOOD].. consumed that you received in-kind over the past 12 months (wages for work, etc.)? IF NONE WRITE ZERO
	NO	YES	CODE	MONTHS	QUANTITY	UNIT	RUPEES	MONTHS	QUANTITY	UNIT	RUPEES	RUPEES
1. GRAINS AND CEREALS:			**010**									
Fine rice			011									
Coarse rice			012									
Beaten, flattened rice			013									
Maize			014									
Maize flour			015									
Wheat flour			016									
Millet			017									
Other grains/cereals			018									
2. PULSES AND LENTILS:			**020**									
Black Pulse			021									
Masoor			022									
Rahar			023									
Gram			024									
Other pulses			025									
Other beans			026									

FOOD EXPENSES AND HOME PRODUCTION (CONT.)

1. Have you consumed ..[FOOD].. during the past 12 months? PUT A CHECK (✓) IN THE APPROPRIATE BOX FOR EACH FOOD ITEM. IF THE ANSWER TO Q. 1 IS YES, ASK Q. 2-8.				FOOD PURCHASES: 2. How many months in the past 12 months did you purchase ..[FOOD].. ? IF NONE WRITE ZERO AND ➔5	3. In a typical month during which you purchased ..[FOOD]. how much did you purchase?		4. How much would you normally have to spend in total to buy this quantity?	HOME PRODUCTION: 5. How many months in the past 12 months did you consume ..[FOOD].. that you grew or produced yourself? IF NONE WRITE ZERO AND ➔8	6. In a typical month during which you ate ..[FOOD].., how much did your household consume of ..[FOOD]..?		7. How much would your household have to spend in the market to buy this quantity of ..[FOOD].. (i.e. the amount consumed in a typical month)?	IN-KIND: 8. What is the total value of the ..[FOOD].. consumed that you received in-kind over the past 12 months (wages for work, etc.)? IF NONE WRITE ZERO
	NO	YES	CODE	MONTHS	QUANTITY	UNIT	RUPEES	MONTHS	QUANTITY	UNIT	RUPEES	RUPEES
3. EGGS AND MILK PRODUCTS			**030**									
Eggs			031									
Milk			032									
Condensed milk			033									
Baby milk/powder milk			034									
Curd			035									
Other milk products			036									
4. COOKING OILS			**040**									
Ghee			041									
Vegetable oil			042									
Mustard oil			043									
Other oil			044									
5. VEGETABLES:			**050**									
Potatoes/pindaaloo			051									
Onions			052									
Cauliflower/cabbage			053									
Tomatoes			054									
Green leafy vegetables			055									
Other vegetables			056									

1. Have you consumed ..[FOOD].. during the past 12 months? PUT A CHECK (✓) IN THE APPROPRIATE BOX FOR EACH FOOD ITEM. IF THE ANSWER TO Q. 1 IS YES, ASK Q. 2-8.				FOOD PURCHASES				HOME PRODUCTION				IN-KIND
				2. How many months in the past 12 months did you purchase ..[FOOD].. ? IF NONE WRITE ZERO AND ➔5	3. In a typical month during which you purchased ..[FOOD]. how much did you purchase?		4. How much would you normally have to spend in total to buy this quantity?	5. How many months in the past 12 months did you consume ..[FOOD].. that you grew or produced yourself? IF NONE WRITE ZERO AND ➔8	6. In a typical month during which you ate ..[FOOD].., how much did your household consume of ..[FOOD]..?		7. How much would your household have to spend in the market to buy this quantity of ..[FOOD].. (i.e. the amount consumed in a typical month)?	8. What is the total value of the ..[FOOD].. consumed that you received in-kind over the past 12 months (wages for work, etc.)? IF NONE WRITE ZERO
	NO	YES	CODE	MONTHS	QUANTITY	UNIT	RUPEES	MONTHS	QUANTITY	UNIT	RUPEES	RUPEES
6. FRUITS AND NUTS:			**060**									
Bananas			061									
Citrus fruits (oranges, etc.)			062									
Mangoes			063									
Apples			064									
Pineapple			065									
Papaya			066									
Other fruits			067									
Dried fruits			068									
7. FISH AND MEAT:			**070**									
Fish			071									
Mutton			072									
Buff.			073									
Chicken			074									
Other meats (boar, duck, etc.)			075									

FOOD EXPENSES AND HOME PRODUCTION (CONT.)

1. Have you consumed ..[FOOD].. during the past 12 months? PUT A CHECK (✓) IN THE APPROPRIATE BOX FOR EACH FOOD ITEM. IF THE ANSWER TO Q. 1 IS YES, ASK Q. 2-8.				FOOD PURCHASES: 2. How many months in the past 12 months did you purchase ..[FOOD].. ? IF NONE WRITE ZERO AND ➔5	FOOD PURCHASES: 3. In a typical month during which you purchased ..[FOOD]. how much did you purchase?		FOOD PURCHASES: 4. How much would you normally have to spend in total to buy this quantity?	HOME PRODUCTION: 5. How many months in the past 12 months did you consume ..[FOOD].. that you grew or produced yourself? IF NONE WRITE ZERO AND ➔8	HOME PRODUCTION: 6. In a typical month during which you ate ..[FOOD].., how much did your household consume of ..[FOOD]..?		HOME PRODUCTION: 7. How much would your household have to spend in the market to buy this quantity of ..[FOOD].. (i.e. the amount consumed in a typical month)?	IN-KIND: 8. What is the total value of the ..[FOOD].. consumed that you received in-kind over the past 12 months (wages for work, etc.)? IF NONE WRITE ZERO
	NO	YES	CODE	MONTHS	QUANTITY	UNIT	RUPEES	MONTHS	QUANTITY	UNIT	RUPEES	RUPEES
8. SPICES AND CONDIMENTS:			**080**									
Salt			081									
Cumin seed/black pepper			082									
Turmeric			083									
Ginger and garlic			084									
Chilies			085									
Other spices and condiments			086									
9. SWEETS AND CONFECTIONERY:			**090**									
Sugar			091									
Gur			092									
Sweets (mithai)			093									
Sugar candy, chocolate, etc.			094									
10. NON-ALCOHOLIC BEVERAGES			**100**									
Tea (dried leaves)			101									
Coffee (ground, instant)			102									
Carbonated drinks, fruit juices			103									
Other non-alcoholic drinks			104									

FOOD EXPENSES AND HOME PRODUCTION (CONT.)

1. Have you consumed ..[FOOD].. during the past 12 months? PUT A CHECK (✓) IN THE APPROPRIATE BOX FOR EACH FOOD ITEM. IF THE ANSWER TO Q. 1 IS YES, ASK Q. 2-8.				FOOD PURCHASES: 2. How many months in the past 12 months did you purchase ..[FOOD].. ? IF NONE WRITE ZERO AND ➔5	FOOD PURCHASES: 3. In a typical month during which you purchased ..[FOOD]. how much did you purchase?		FOOD PURCHASES: 4. How much would you normally have to spend in total to buy this quantity?	HOME PRODUCTION: 5. How many months in the past 12 months did you consume ..[FOOD].. that you grew or produced yourself? IF NONE WRITE ZERO AND ➔8	HOME PRODUCTION: 6. In a typical month during which you ate ..[FOOD].., how much did your household consume of ..[FOOD]..?		HOME PRODUCTION: 7. How much would your household have to spend in the market to buy this quantity of ..[FOOD].. (i.e. the amount consumed in a typical month)?	IN-KIND: 8. What is the total value of the ..[FOOD].. consumed that you received in-kind over the past 12 months (wages for work, etc.)? IF NONE WRITE ZERO
	NO	YES	CODE	MONTHS	QUANTITY	UNIT	RUPEES	MONTHS	QUANTITY	UNIT	RUPEES	RUPEES
11. ALCOHOLIC BEVERAGES:			**110**									
Wine			111									
Gin, whiskey			112									
Beer/jandh			113									
Other alcoholic drinks			114									
12. TOBACCO & TOBACCO PRODUCTS:			**120**									
Cigarettes			121									
Bidis			122									
Tobacco			123									
Other (jarda, khaini, betel nut)			124									
13. MISC. FOOD PRODUCTS:			**130**									
Meals taken outside home			131									
Misc. other food expenditures			132									

ASK RESPONDENT TO ESTIMATE AVERAGE MONTHLY EXPENDITURE ON FOOD, VALUE OF HOME PRODUCED FOOD, AND FOOD RECEIVED IN KIND 140

B Code

code
[11pt]article
[T1]fontenc mathpazo
graphicx caption nolabel
adjustbox xcolor enumerate geometry amsmath amssymb textcomp upquote eurosym [mathletters]ucs [utf8x]inputenc fancyvrb grffile hyperref longtable booktabs [inline]enumitem [normalem]ulem mathrsfs
HighlightingVerbatimcommandchars=
{}
verbose,tmargin=1in,bmargin=1in,lmargin=1in,rmargin=1in
[commandchars=
{}] In [1]: import os import random import sys import json
import numpy as np import pandas as pd from pandas.io.stata import StataReader
from matplotlib import pyplot as plt from IPython.display import display import seaborn as sns sns.set()
from sklearn.model˙selection import train˙test˙split
from sklearn.cluster import KMeans from sklearn.preprocessing import MinMaxScaler from sklearn.cluster import SpectralClustering
#from apyori import apriori from mlxtend.frequent˙patterns import apriori from mlxtend.frequent˙patterns import association˙rules
Initial Exploration and some cleaning
[commandchars=
{}] In [2]: food˙nlss1 = pd.read˙stata(”NLSS1/Z05a.dta”,convert˙categoricals=True).fillna(0.00) print(”total unique households = ” + str(len(food˙nlss1['WWWHH'].unique()))) print(”total unique items = ” + str(len(food˙nlss1['S05A˙ITM'].unique())))
[commandchars=
{}] total unique households = 3373 total unique items = 67
[commandchars=
{}] In [3]: food˙nlss1.drop('WWW',axis=1,inplace=True) #drop VDC code food˙nlss1.drop('HH',axis=1,inplace=True) #drop Household code
food˙nlss1.head()
[commandchars=
{}] Out[3]: WWWHH S05A_ITM S05A_02 S05A_03A S05A_03B S05A_04 S05A_05 S05A_06A \ 0 101.0 12 4.0 75.0 1.0 900.0 8.0 75.0 1 101.0 14 0.0 0.0 0.0 0.0 1.0 15.0 2 101.0 16 0.0 0.0 0.0 0.0 6.0 25.0 3 101.0 22 0.0 0.0 0.0 0.0 12.0 5.0 4 101.0 31 0.0 0.0 0.0 0.0 2.0 21.0
S05A_06B S05A_07 S05A_08 0 1.0 900.0 0.0 1 1.0 105.0 0.0 2 1.0 250.0 0.0 3 1.0 115.0 0.0 4 9.0 63.0 0.0
Unique food Items and some popular Items
[commandchars=
{}] In [4]: df˙explore = food˙nlss1
df˙explore['S05A˙ITM'] = df˙explore['S05A˙ITM'].astype('category')
food˙category˙purchase = df˙explore[['S05A˙04','S05A˙ITM']]“ .groupby('S05A˙ITM')['S05A˙04'] “ .mean()

food˙category˙purchase.index = ['Fine rice','Coarse rice','Beaten/Flattened rice' ,'Maize','Maize flour',“ 'Wheat flour','Millet','Other grains/cerials','Black pulse','Masoor','Rahar','Gram',“ 'Other pulses','Other beans','Eggs','Milk','Condensed milk','Baby milk/powder milk',“ 'Curd','Other milk Products','Ghee','vegetable oil','Mustard oil','Other oil',“ 'Potatoes/Sweet Potatoes','Onions','Cauliflower/Cabbage','Tomatoes',“ 'Greeen leafy Vegetables','Other Vegetables','Bananas','Citrus Fruits','Mangoes',“ 'Apples','Pineapple','Papaya','Other fruits','Dried fruits','Fish','Mutton',“ 'Buffalo meat','Chicken','Other meats','Salt','Cumin seed/black pepper',“ 'turmeric','Ginger and Garlic','chilies','Other spices and condements','Sugar',“ 'Caramel','Sweets','Sugar Candy','Tea','Coffee','Carbonated drinks',“ 'Other non-alcoholic drinks','Wine','Gin, Whiskey','Beer','Other alcoholic drinks',“ 'Cigarettes','Bidis','Tobacco','Other tobaco products',“ 'Meals taken Outside home','other food expences']

food˙purchase˙dict = food˙category˙purchase.sort˙values(ascending=**False**).to˙dict()

food˙category˙grown = df˙explore[['S05A˙07','S05A˙ITM']]“ .groupby('S05A˙ITM')['S05A˙07'] “ .mean()

food˙category˙grown.index = ['Fine rice','Coarse rice','Beaten/Flattened rice' ,'Maize','Maize flour',“ 'Wheat flour','Millet','Other grains/cerials','Black pulse','Masoor','Rahar','Gram',“ 'Other pulses','Other beans','Eggs','Milk','Condensed milk','Baby milk/powder milk',“ 'Curd','Other milk Products','Ghee','vegetable oil','Mustard oil','Other oil',“ 'Potatoes/Sweet Potatoes','Onions','Cauliflower/Cabbage','Tomatoes',“ 'Greeen leafy Vegetables','Other Vegetables','Bananas','Citrus Fruits','Mangoes',“ 'Apples','Pineapple','Papaya','Other fruits','Dried fruits','Fish','Mutton',“ 'Buffalo meat','Chicken','Other meats','Salt','Cumin seed/black pepper',“ 'turmeric','Ginger and Garlic','chilies','Other spices and condements','Sugar',“ 'Caramel','Sweets','Sugar Candy','Tea','Coffee','Carbonated drinks',“ 'Other non-alcoholic drinks','Wine','Gin, Whiskey','Beer','Other alcoholic drinks',“ 'Cigarettes','Bidis','Tobacco','Other tobaco products',“ 'Meals taken Outside home','other food expences']

food˙growth˙dict = food˙category˙grown.sort˙values(ascending=**False**).to˙dict()

food˙names = list(food˙purchase˙dict.keys())[:20] count˙values = list(food˙purchase˙dict.values())[:20]

food˙names˙g = list(food˙growth˙dict.keys())[:20] count˙values˙g = list(food˙growth˙dict.values())[:20]

def plot˙bar˙graph(x,y,title): fig, axs = plt.subplots(1, 1, figsize=(20, 10), sharey=**True**) axs.bar(x, y) axs.set˙title(title) plt.xticks(rotation =90)

 # data labels **for** i, v **in** enumerate(y): axs.text(i-.25, v+10, y[i], fontsize=8, *#color=label˙color˙list[i]*) **return** plt.show()

[commandchars=
{}] In [5]: **def** plot˙bar˙graph(x,y,title): fig, axs = plt.subplots(1, 1, figsize=(20, 10), sharey=**True**) axs.bar(x, y) axs.set˙title(title) plt.xticks(rotation =90)

 # data labels **for** i, v **in** enumerate(y): axs.text(i-.25, v+10, y[i], fontsize=8, *#color=label˙color˙list[i]*) **return** plt.show()

[commandchars=
{}] In [6]: plot˙bar˙graph(food˙names,count˙values,"Top 20 Food items purchased ordered by average spending")

max size=0.90.9output$_{70}$.*png*

[commandchars=

{}] In [7]: plot˙bar˙graph(food˙names˙g,count˙values˙g,"Top 20 Food items purchased ordered by average of possible spending")

max size=0.90.9output$_{80}$.*png*

[commandchars=
{}] In [8]: food˙nlss1 = pd.read˙stata("NLSS1/Z05a.dta",convert˙categoricals=**True**).fillna(0.00) food˙nlss1.drop('WWW',axis=1,inplace=**True**) food˙nlss1.drop('HH',axis=1,inplace=**True**) food˙nlss1.drop('S05A˙02',axis=1,inplace=**True**) food˙nlss1.drop('S05A˙04',axis=1,inplace=**True**) food˙nlss1.drop('S05A˙05',axis=1,inplace=**True**) food˙nlss1.drop('S05A˙07',axis=1,inplace=**True**) food˙nlss1.drop('S05A˙08',axis=1,inplace=**True**)

[commandchars=
{}] In [9]: food˙nlss1.head(5)

[commandchars=
{}] Out[9]: WWWHH S05A_ITM S05A_03A S05A_03B S05A_06A S05A_06B 0 101.0 12 75.0 1.0 75.0 1.0 1 101.0 14 0.0 0.0 15.0 1.0 2 101.0 16 0.0 0.0 25.0 1.0 3 101.0 22 0.0 0.0 5.0 1.0 4 101.0 31 0.0 0.0 21.0 9.0

Exploration and data processing

[commandchars=
{}] In [10]: food˙nlss1.columns

[commandchars=
{}] Out[10]: Index(['WWWHH', 'S05A_ITM', 'S05A_03A', 'S05A_03B', 'S05A_06A', 'S05A_06B'], dtype='object')

[commandchars=
{}] In [11]: food˙nlss1.columns=['WWWHH','item','amount˙purchased','unit˙purchased','amount˙grown','unit˙grown']

[commandchars=
{}] In [12]: food˙nlss1.head(5)

[commandchars=
{}] Out[12]: WWWHH item amount_purchased unit_purchased amount_grown unit_grown 0 101.0 12 75.0 1.0 75.0 1.0 1 101.0 14 0.0 0.0 15.0 1.0 2 101.0 16 0.0 0.0 25.0 1.0 3 101.0 22 0.0 0.0 5.0 1.0 4 101.0 31 0.0 0.0 21.0 9.0

Data Frame For K-means

[commandchars=
{}] In [13]: *#convert all units in grams* l1 = [] **for** i **in** food˙nlss1['unit˙purchased']: **if** i == 0.0: l1.append(0.0) **if** i == 1.0: l1.append(1.0e3) **if** i == 2.0: l1.append(1.0) **if** i == 3.0: l1.append(37.3242*1.0e3) **if** i == 4.0: l1.append(1.0e3) **if** i == 5.0: l1.append(87215*1.0e3) **if** i == 6.0: l1.append(4361*1.0e3) **if** i == 7.0: l1.append(0.545*1.0e3) **if** i == 8.0: l1.append(1.0e3) **if** i == 9.0: l1.append(100.0) **if** i == 10.0: l1.append(500.0) food˙nlss1['unit˙purchased']= l1

[commandchars=
{}] In [14]: l2 = [] **for** i **in** food˙nlss1['unit˙grown']: **if** i == 0.0: l2.append(0.0) **if** i == 1.0: l2.append(1.0e3) **if** i == 2.0: l2.append(1.0) **if** i == 3.0: l2.append(37.3242*1.0e3) **if** i == 4.0: l2.append(1.0e3) **if** i == 5.0: l2.append(87215*1.0e3) **if** i == 6.0: l2.append(4361*1.0e3) **if** i == 7.0: l2.append(0.545*1.0e3) **if** i == 8.0: l2.append(1.0e3) **if** i == 9.0: l2.append(100.0) **if** i == 10.0: l2.append(500.0) food˙nlss1['unit˙grown']= l2

[commandchars=
{}] In [15]: food˙nlss1.head(5)

[commandchars=
{}] Out[15]: WWWHH item amount_purchased unit_purchased amount_grown unit_grown 0 101.0 12 75.0 1000.0 75.0 1000.0 1 101.0 14 0.0 0.0 15.0 1000.0 2 101.0 16 0.0 0.0 25.0 1000.0 3 101.0 22 0.0 0.0 5.0 1000.0 4 101.0 31 0.0 0.0 21.0 100.0

[commandchars=
{}] In [16]: food˙nlss1['consumption˙purchase']=food˙nlss1.apply(**lambda** row: row['amount˙purchased']*row['unit˙purchased'], axis=1) food˙nlss1['consumption˙grown']=food˙nlss1.apply(**lambda** row: row['amount˙grown']*row['unit˙grown'], axis=1)

Gram˙intake = food˙nlss1

[commandchars=
{}] In [17]: food˙nlss1.head(5)

[commandchars=
{}] Out[17]: WWWHH item amount_purchased unit_purchased amount_grown unit_grown \ 0 101.0 12 75.0 1000.0 75.0 1000.0 1 101.0 14 0.0 0.0 15.0 1000.0 2 101.0 16 0.0 0.0 25.0 1000.0 3 101.0 22 0.0 0.0 5.0 1000.0 4 101.0 31 0.0 0.0 21.0 100.0

consumption_purchase consumption_grown 0 75000.0 75000.0 1 0.0 15000.0 2 0.0 25000.0 3 0.0 5000.0 4 0.0 2100.0

[commandchars=
{}] In [98]: *#total consumption of food groups in grams*

 # Grain/Cerial grains˙ = pd.DataFrame(food˙nlss1[food˙nlss1.item ¿= 11]“ [food˙nlss1[food˙nlss1.item ¿= 11].item ¡= 18]“ .set˙index(['WWWHH'])) grain˙consumption = pd.DataFrame(pd.DataFrame(grains˙[['consumption˙purchase','consumption˙grown']]“ .groupby('WWWHH')[['consumption˙purchase','consumption˙grown']].sum())“ [['consumption˙purchase','consumption˙grown']].sum(axis=1),columns=['grains˙cerials']) *#grain˙consumption['grains/cerials']=grain˙consumption.apply(lambda row: row['grains/cerials']*4e-3, axis=1)*

#Pulses/Lentals pulses˙ = pd.DataFrame(food˙nlss1[food˙nlss1.item ¿= 21]“ [food˙nlss1[food˙nlss1.item ¿= 21].item ¡= 26]“ .set˙index(['WWWHH'])) pulses˙consumption = pd.DataFrame(pd.DataFrame(pulses˙[['consumption˙purchase','consumption˙grown']]“ .groupby('WWWHH')[['consumption˙purchase','consumption˙grown']].sum())“ [['consumption˙purchase','consumption˙grown']].sum(axis=1),columns=['pulses˙lentals']) *#pulses˙consumption['pulses/lentals']=pulses˙consumption.apply(lambda row: row['pulses/lentals']*4e-3, axis=1)*

 # poultry/Dairy dairy˙ = pd.DataFrame(food˙nlss1[food˙nlss1.item ¿= 31]“ [food˙nlss1[food˙nlss1.item ¿= 31].item ¡= 36]“ .set˙index(['WWWHH'])) dairy˙consumption = pd.DataFrame(pd.DataFrame(dairy˙[['consumption˙purchase','consumption˙grown']]“ .groupby('WWWHH')[['consumption˙purchase','consumption˙grown']].sum())“ [['consumption˙purchase','consumption˙grown']].sum(axis=1),columns=['dairy']) *#dairy˙consumption['poultry/dairy']=dairy˙consumption.apply(lambda row: row['poultry/dairy']*9e-3, axis=1)*

#Oil/Fat

oil˙ = pd.DataFrame(food˙nlss1[food˙nlss1.item ¿= 41]“ [food˙nlss1[food˙nlss1.item ¿= 41].item ¡= 44]“ .set˙index(['WWWHH'])) oil˙consumption = pd.DataFrame(pd.DataFrame(oil˙[['consumption˙purchase','consumption˙grown']]“ .groupby('WWWHH')[['consumption˙purchase','consumption˙grown']].sum())“ [['consumption˙purchase','consumption˙grown']].sum(axis=1),columns=['oil˙fat'])

#oil˙consumption['oil/fat']=oil˙consumption.apply(lambda row: row['oil/fat']*9e-3, axis=1)

#Vegetables vegetables˙ = pd.DataFrame(food˙nlss1[food˙nlss1.item ¿= 51]“ [food˙nlss1[food˙nlss1.item ¿= 51].item ¡= 56]“ .set˙index(['WWWHH'])) vegetables˙consumption = pd.DataFrame(pd.DataFrame(vegetables˙[['consumption˙purchase','consumption˙grown']]“ .groupby('WWWHH')[['consumption˙purchase','consumption˙grown']].sum())“ [['consumption˙purchase','consumption˙grown']].sum(axis=1),columns=['vegetables']) #vegetables˙consumption['vegetables']=vegetables˙consumption.apply(lambda row: row['vegetables']*4e-3, axis=1)

#Fruits fruits˙ = pd.DataFrame(food˙nlss1[food˙nlss1.item ¿= 61]“ [food˙nlss1[food˙nlss1.item ¿= 61].item ¡= 68]“ .set˙index(['WWWHH'])) fruits˙consumption = pd.DataFrame(pd.DataFrame(fruits˙[['consumption˙purchase','consumption˙grown']]“ .groupby('WWWHH')[['consumption˙purchase','consumption˙grown']].sum())“ [['consumption˙purchase','consumption˙grown']].sum(axis=1),columns=['fruits'])
#fruits˙consumption['fruits']=fruits˙consumption.apply(lambda row: row['fruits']*4e-3, axis=1)

#Meat/Fish meat˙ = pd.DataFrame(food˙nlss1[food˙nlss1.item ¿= 71]“ [food˙nlss1[food˙nlss1.item ¿= 71].item ¡= 75]“ .set˙index(['WWWHH'])) meat˙consumption = pd.DataFrame(pd.DataFrame(meat˙[['consumption˙purchase','consumption˙grown']]“ .groupby('WWWHH')[['consumption˙purchase','consumption˙grown']].sum())“ [['consumption˙purchase','consumption˙grown']].sum(axis=1),columns=['meat˙fish'])
#meat˙consumption['meat/fish']=meat˙consumption.apply(lambda row: row['meat/fish']*4e-3, axis=1)

#Sugar sugar˙ = pd.DataFrame(food˙nlss1[food˙nlss1.item ¿= 91]“ [food˙nlss1[food˙nlss1.item ¿= 91].item ¡= 94]“ .set˙index(['WWWHH'])) sugar˙consumption = pd.DataFrame(pd.DataFrame(sugar˙[['consumption˙purchase','consumption˙grown']]“ .groupby('WWWHH')[['consumption˙purchase','consumption˙grown']].sum())“ [['consumption˙purchase','consumption˙grown']].sum(axis=1),columns=['sugar'])
#sugar˙consumption['sugar']=sugar˙consumption.apply(lambda row: row['sugar']*9e-3, axis=1)

#Merge all these individual Datasets df˙consumption = pd.merge(grain˙consumption, pulses˙consumption, “ left˙on = 'WWWHH', right˙on = 'WWWHH') df˙consumption = pd.merge(df˙consumption, dairy˙consumption, “ left˙on = 'WWWHH', right˙on = 'WWWHH') df˙consumption = pd.merge(df˙consumption, oil˙consumption, “ left˙on = 'WWWHH', right˙on = 'WWWHH') df˙consumption = pd.merge(df˙consumption, vegetables˙consumption, “ left˙on = 'WWWHH', right˙on = 'WWWHH') df˙consumption = pd.merge(df˙consumption, fruits˙consumption, “ left˙on = 'WWWHH', right˙on = 'WWWHH') df˙consumption = pd.merge(df˙consumption, meat˙consumption, “ left˙on = 'WWWHH', right˙on = 'WWWHH') df˙consumption = pd.merge(df˙consumption, sugar˙consumption, “ left˙on = 'WWWHH', right˙on = 'WWWHH')

df = df˙consumption

[commandchars=
{}] In [19]: df˙consumption.head(5)

def plot˙bar˙graph2(x,y,title): fig, axs = plt.subplots(1, 1, figsize=(20, 10), sharey=**True**) axs.bar(x, y) axs.set˙title(title) plt.xticks(rotation =90)

 # data labels **for** i, v **in** enumerate(y): axs.text(i-.25, v+10, y[i], fontsize=8, *#color=label˙color˙list[i]*) **return** plt.show()

[commandchars=
{}] In [21]: df˙dic = df˙consumption.mean(axis=0)

list11˙=[] for i in range (0,df˙dic.shape[0]): # Create list for the current row
list11˙.append(df˙dic[i]) list11˙
inc = ['grains˙cerials', 'pulses˙lentals', 'poultry˙dairy', 'oil˙fat', 'vegetables', 'fruits', 'meat˙fish', 'sugar']
fig = plt.figure() ax0 = plt.subplot(1,2,1)
explode = (0.1,0.1,0.1,0.1,0.1,0.1,0.1,0.1)
wedges, texts, autotexts = ax0.pie(list11˙,explode=explode, autopct='%1.1f%%', textprops=dict(color="k"))
plt.setp(autotexts, size=12, weight="bold")
ax0.set˙title('Percentage share of average monetary value for home-grown food-types '“ ,y=1,fontsize=15,fontweight='bold') ax0.legend(wedges,inc, loc="best", title = 'Food-types', bbox˙to˙anchor=(1, 0, 0.5, 1))
#Set the figure size so that all four panel graphs are clearly visible fig.set˙size˙inches(20,10) *#Set a title for the figure* fig.suptitle('Perc',y=1,fontsize=20,fontweight='bold') plt.show()

max size=0.90.9output$_2$3$_0$*.png*

Kmeans
[commandchars=
{}] In [103]: from scipy.spatial.distance import cdist *# k means determine k* distortions = [] K = range(1,15) for k in K: kmeanModel = KMeans(n˙clusters=k).fit(df˙consumption) kmeanModel.fit(df˙consumption) distortions.append(sum(np.min(cdist(df˙consumption, kmeanModel.cluster˙centers˙, “ 'euclidean'), axis=1)) / df˙consumption.shape[0]) y0 = np.linspace(0, 1000,1000) x0= 3*np.ones(1000) y = np.linspace(0, 1000,1000) x= 5*np.ones(1000) y2 = np.linspace(0, 1000,1000) x2= 8*np.ones(1000) *# Plot the elbow* ax = plt.subplot() ax.plot(K, distortions, 'bx-') axx = ax.twinx() axx.plot(x,y,"--r") axx.plot(x2,y2,"--y") axx.plot(x0,y0,"--g") ax.set˙xlabel('k') ax.grid(False) axx.grid(False) ax.set˙ylabel('Distortion') ax.set˙title('The Elbow Method showing the optimal k') plt.show()

max size=0.90.9output$_2$5$_0$*.png*

[commandchars=
{}] In [41]: *#K-means k=5* random.seed(10) scaler = MinMaxScaler() ddf˙scaled = scaler.fit˙transform(df˙consumption) kmeans = KMeans(n˙clusters=5, max˙iter=600, init ='k-means++', algorithm = 'auto') kmeans.fit(ddf˙scaled)
[commandchars=
{}] Out[41]: KMeans(algorithm='auto', copy_x=True, init='k-means++', max_iter=600, n_clusters=5, n_init=10, n_jobs=None, precompute_distances='auto', random_state=None, tol=0.0001, verbose=0)
[commandchars=
{}] In [42]: df˙consumption['total'] = df˙consumption.sum(axis=1)
def percentage˙share(df): for i in df.columns: df[i] = df.apply(lambda row: (row[i]/row['total']*100), axis=1) return df
df˙consumption = percentage˙share(df˙consumption) df˙consumption.drop('total',axis=1,inplace=True) df˙consumption['cluster˙label']= kmeans.predict(ddf˙scaled)
df˙c = df˙consumption.set˙index('cluster˙label') consumption˙share˙cluster = df˙c.groupby('cluster˙label')[df˙c.columns].mean() consumption˙share˙cluster.columns =

['grains˙cerials', 'pulses˙lentals', 'poultry˙dairy', 'oil˙fat', 'vegetables', 'fruits', 'meat˙fish', 'sugar'] consumption˙share˙cluster.sort˙values(["fruits"],ascending=True)

[commandchars=
{}] /anaconda3/lib/python3.7/site-packages/ipykernel_launcher.py:5: RuntimeWarning: invalid value encountered in double_scalars """

[commandchars=
{}] Out[42]: grains_cerials pulses_lentals poultry_dairy oil_fat \ cluster_label 2 86.377041 0.727356 0.019088 0.215470 1 66.076251 1.660805 0.062673 0.001497 3 59.300297 0.971895 38.841942 0.345977 0 73.814927 2.965131 6.920791 1.220846 4 54.475372 4.201936 6.198237 0.832427

vegetables fruits meat_fish sugar cluster_label 2 12.653670 0.003675 0.002659 0.001040 1 32.191960 0.004629 0.001586 0.000600 3 0.528823 0.007749 0.002294 0.001023 0 8.119519 4.203408 1.541554 1.213825 4 8.597867 22.861464 0.895126 1.937572

[commandchars=
{}] In [43]: spend˙nlss1 = pd.read˙stata("NLSS1/Z05a.dta",“ convert˙categoricals=True).fillna(0.00) spend˙nlss1.drop('WWW',axis=1,inplace=True) spend˙nlss1.drop('HH',axis=1,inplace=True) spend˙nlss1.drop('S05A˙02',axis=1,inplace=True) spend˙nlss1.drop('S05A˙03A',axis=1,inplace=True) spend˙nlss1.drop('S05A˙03B',axis=1,inplace=True) spend˙nlss1.drop('S05A˙05',axis=1,inplace=True) spend˙nlss1.drop('S05A˙06A',axis=1,inplace=True) spend˙nlss1.drop('S05A˙06B',axis=1,inplace=True) spend˙nlss1.drop('S05A˙08',axis=1,inplace=True)

food˙spenditure = pd.DataFrame(spend˙nlss1[['WWWHH','S05A˙04']]“ .groupby('WWWHH')['S05A˙04'].sum())

food˙saving = pd.DataFrame(spend˙nlss1[['WWWHH','S05A˙07']]“ .groupby('WWWHH')['S05A˙07'].sum())

df˙expence = pd.merge(food˙spenditure, food˙saving, left˙on = 'WWWHH', right˙on = 'WWWHH') df˙expence.columns=['price˙purchase','price˙growth']

df˙expence['total'] = df˙expence.sum(axis=1)

def percentage˙share(df): **for** i **in** df.columns: df[i] = df.apply(**lambda** row: (row[i]/row['total']*100), axis=1) **return** df

df˙expence = percentage˙share(df˙expence)

df˙expence.drop('total',axis=1,inplace=True)

df˙expence = pd.merge(df˙expence, df˙consumption,“ left˙on = 'WWWHH', right˙on = 'WWWHH') df˙expence['cluster˙label'] = kmeans.predict(ddf˙scaled)

df˙ = df˙expence.set˙index('cluster˙label') expence˙share˙cluster = df˙.groupby('cluster˙label')[df˙.columns].mean() expence˙share˙cluster.drop(['grains˙cerials','pulses˙lentals','dairy','oil˙fat', 'vegetables','fruits','meat˙fish','sugar'],axis=1,inplace=True) expence˙share˙cluster

[commandchars=
{}] /anaconda3/lib/python3.7/site-packages/ipykernel_launcher.py:25: RuntimeWarning: invalid value encountered in double_scalars

[commandchars=
{}] Out[43]: price_purchase price_growth cluster_label 0 58.190376 41.809624 1 38.257028 61.742972 2 40.948213 59.051787 3 30.347886 69.652114 4 42.683533 57.316467

[commandchars=
{}] In [110]: clusters = [0,1,2,3,4] purchase = expence˙share˙cluster.price˙purchase growth = expence˙share˙cluster.price˙growth header = expence˙share˙cluster.columns.values *#an array of columns headers*

fig = plt.figure() ax = plt.subplot()

#Grouped bar chart ind = np.arange(len(clusters)) *#no of x ticks; months on our data* width = 0.3 *#width of the bar*

#ax.bar(position of the bar wrt the x-ticks, data, width of bar, label) ax.bar(ind - width, purchase, width,label='Purchased Items') ax.bar(ind , growth, width,label='Home-Grown Items') ax.legend(loc=2) ax.yaxis.grid(**True**)

for tick **in** ax.get˙xticklabels(): tick.set˙rotation(90)

#Set the figure size so that all four panel graphs are clearly visible fig.set˙size˙inches(10,5) *#Set a title for the figure* fig.suptitle(' Percentage share of average monetary value of purchased " and home-grown items for identified clusters '" ,y=1,fontsize=15,fontweight='bold') plt.show()

max size=0.90.9output$_2$9$_0$.*png*

[commandchars=
{}] In [46]: spend˙nlss1.columns=['WWWHH','item','amount˙purchased','amount˙grown']

 # Grain/Cerial grains˙ = pd.DataFrame(spend˙nlss1[spend˙nlss1.item ¿= 11]" [spend˙nlss1[spend˙nlss1.item ¿= 11].item ¡= 18]" .set˙index(['WWWHH'])) grain˙consumption = pd.DataFrame(pd.DataFrame(grains˙[['amount˙purchased']]" .groupby('WWWHH')[['amount˙purchased']].sum())) grain˙consumption.columns=['grains˙cerials']

#Pulses/Lentals pulses˙ = pd.DataFrame(spend˙nlss1[spend˙nlss1.item ¿= 21]" [spend˙nlss1[spend˙nlss1.item ¿= 21].item ¡= 26]" .set˙index(['WWWHH'])) pulses˙consumption = pd.DataFrame(pd.DataFrame(pulses˙[['amount˙purchased']]" .groupby('WWWHH')[['amount˙purchased']].sum())) pulses˙consumption.columns=['pulses˙lentals']

 # poultry/Dairy dairy˙ = pd.DataFrame(spend˙nlss1[spend˙nlss1.item ¿= 31]" [spend˙nlss1[spend˙nlss1.item ¿= 31].item ¡= 36]" .set˙index(['WWWHH'])) dairy˙consumption = pd.DataFrame(pd.DataFrame(dairy˙[['amount˙purchased']]" .groupby('WWWHH')[['amount˙purchased']].sum())) dairy˙consumption.columns=['dairy']

#Oil/Fat

oil˙ = pd.DataFrame(spend˙nlss1[spend˙nlss1.item ¿= 41]" [spend˙nlss1[spend˙nlss1.item ¿= 41].item ¡= 44]" .set˙index(['WWWHH'])) oil˙consumption = pd.DataFrame(pd.DataFrame(oil˙[['amount˙purchased']]" .groupby('WWWHH')[['amount˙purchased']].sum())) oil˙consumption.columns=['oil˙fat']

#Vegetables vegetables˙ = pd.DataFrame(spend˙nlss1[spend˙nlss1.item ¿= 51]" [spend˙nlss1[spend˙nlss1.item ¿= 51].item ¡= 56]" .set˙index(['WWWHH'])) vegetables˙consumption = pd.DataFrame(pd.DataFrame(vegetables˙[['amount˙purchased']]" .groupby('WWWHH')[['amount˙purchased']].sum())) vegetables˙consumption.columns=['vegetables']

#Fruits fruits˙ = pd.DataFrame(spend˙nlss1[spend˙nlss1.item ¿= 61]" [spend˙nlss1[spend˙nlss1.item ¿= 61].item ¡= 68]" .set˙index(['WWWHH'])) fruits˙consumption =pd.DataFrame(pd.DataFrame(fruits˙[['amount˙purchased']]" .groupby('WWWHH')[['amount˙purchased']].sum())) fruits˙consumption.columns=['fruits']

#Meat/Fish meat˙ = pd.DataFrame(spend˙nlss1[spend˙nlss1.item ¿= 71]" [spend˙nlss1[spend˙nlss1.item ¿= 71].item ¡= 75]" .set˙index(['WWWHH'])) meat˙consumption = pd.DataFrame(pd.DataFrame(meat˙[['amount˙purchased']]" .groupby('WWWHH')[['amount˙purchased']].sum())) meat˙consumption.columns=['meat˙fish'] *#Sugar* sugar˙ = pd.DataFrame(spend˙nlss1[spend˙nlss1.item ¿= 91]" [spend˙nlss1[spend˙nlss1.item ¿= 91].item ¡= 94]" .set˙index(['WWWHH'])) sugar˙consumption = pd.DataFrame(pd.DataFrame(sugar˙[['amount˙purchased']]" .groupby('WWWHH')[['amount˙purchased']].sum())) sugar˙consumption.columns=['sugar']

#Merge all these individual Datasets df˙purchase = pd.merge(grain˙consumption, pulses˙consumption,“ left˙on = 'WWWHH', right˙on = 'WWWHH') df˙purchase = pd.merge(df˙purchase, dairy˙consumption,“ left˙on = 'WWWHH', right˙on = 'WWWHH') df˙purchase = pd.merge(df˙purchase, oil˙consumption,“ left˙on = 'WWWHH', right˙on = 'WWWHH') df˙purchase = pd.merge(df˙purchase, vegetables˙consumption,“ left˙on = 'WWWHH', right˙on = 'WWWHH') df˙purchase = pd.merge(df˙purchase, fruits˙consumption,“ left˙on = 'WWWHH', right˙on = 'WWWHH') df˙purchase = pd.merge(df˙purchase, meat˙consumption,“ left˙on = 'WWWHH', right˙on = 'WWWHH') df˙purchase = pd.merge(df˙purchase, sugar˙consumption,“ left˙on = 'WWWHH', right˙on = 'WWWHH') df˙purchase['total'] = df˙purchase.sum(axis=1) df˙purchase = percentage˙share(df˙purchase) df˙purchase.drop('total',axis=1,inplace=**True**) df˙purchase['cluster˙label'] = kmeans.predict(ddf˙scaled) df˙p = df˙purchase.set˙index('cluster˙label')

p˙share˙cluster = df˙p.groupby('cluster˙label')[df˙p.columns].mean()

[commandchars=
{}] /anaconda3/lib/python3.7/site-packages/ipykernel_launcher.py:25: RuntimeWarning: invalid value encountered in double_scalars

[commandchars=
{}] In [47]: spend˙nlss1.columns=['WWWHH','item','amount˙purchased','amount˙grown']

 # Grain/Cerial grains˙ = pd.DataFrame(spend˙nlss1[spend˙nlss1.item ¿= 11]“ [spend˙nlss1[spend˙nlss1.item ¿= 11].item ¡= 18]“ .set˙index(['WWWHH'])) grain˙consumption = pd.DataFrame(pd.DataFrame(grains˙[['amount˙grown']]“ .groupby('WWWHH')[['amount˙grown']].sum())) grain˙consumption.columns=['grains˙cerials']

#Pulses/Lentals pulses˙ = pd.DataFrame(spend˙nlss1[spend˙nlss1.item ¿= 21]“ [spend˙nlss1[spend˙nlss1.item ¿= 21].item ¡= 26]“ .set˙index(['WWWHH'])) pulses˙consumption = pd.DataFrame(pd.DataFrame(pulses˙[['amount˙grown']]“ .groupby('WWWHH')[['amount˙grown']].sum())) pulses˙consumption.columns=['pulses˙lentals']

 # poultry/Dairy dairy˙ = pd.DataFrame(spend˙nlss1[spend˙nlss1.item ¿= 31]“ [spend˙nlss1[spend˙nlss1.item ¿= 31].item ¡= 36]“ .set˙index(['WWWHH'])) dairy˙consumption = pd.DataFrame(pd.DataFrame(dairy˙[['amount˙grown']]“ .groupby('WWWHH')[['amount˙grown']].sum())) dairy˙consumption.columns=['dairy']

#Oil/Fat

oil˙ = pd.DataFrame(spend˙nlss1[spend˙nlss1.item ¿= 41]“ [spend˙nlss1[spend˙nlss1.item ¿= 41].item ¡= 44]“ .set˙index(['WWWHH'])) oil˙consumption = pd.DataFrame(pd.DataFrame(oil˙[['amount˙grown']]“ .groupby('WWWHH')[['amount˙grown']].sum())) oil˙consumption.columns=['oil˙fat']

#Vegetables vegetables˙ = pd.DataFrame(spend˙nlss1[spend˙nlss1.item ¿= 51]“ [spend˙nlss1[spend˙nlss1.item ¿= 51].item ¡= 56]“ .set˙index(['WWWHH'])) vegetables˙consumption = pd.DataFrame(pd.DataFrame(vegetables˙[['amount˙grown']]“ .groupby('WWWHH')[['amount˙grown']].sum())) vegetables˙consumption.columns=['vegetables']

#Fruits fruits˙ = pd.DataFrame(spend˙nlss1[spend˙nlss1.item ¿= 61]“ [spend˙nlss1[spend˙nlss1.item ¿= 61].item ¡= 68]“ .set˙index(['WWWHH'])) fruits˙consumption =pd.DataFrame(pd.DataFrame(fruits˙[['amount˙grown']]“ .groupby('WWWHH')[['amount˙grown']].sum())) fruits˙consumption.columns=['fruits']

#Meat/Fish meat˙ = pd.DataFrame(spend˙nlss1[spend˙nlss1.item ¿= 71]“ [spend˙nlss1[spend˙nlss1.item ¿= 71].item ¡= 75]“ .set˙index(['WWWHH'])) meat˙consumption = pd.DataFrame(pd.DataFrame(meat˙[['amount˙grown']]“

.groupby('WWWHH')[['amount˙grown']].sum())) meat˙consumption.columns=['meat˙fish'] #Sugar sugar˙ = pd.DataFrame(spend˙nlss1[spend˙nlss1.item ¿= 91]“ [spend˙nlss1[spend˙nlss1.item ¿= 91].item ¡= 94]“ .set˙index(['WWWHH'])) sugar˙consumption = pd.DataFrame(pd.DataFrame(sugar˙[['amount˙grown']]“ .groupby('WWWHH')[['amount˙grown']].sum())) sugar˙consumption.columns=['sugar']

#Merge all these individual Datasets df˙growth = pd.merge(grain˙consumption, pulses˙consumption,“ left˙on = 'WWWHH', right˙on = 'WWWHH') df˙growth = pd.merge(df˙growth, dairy˙consumption,“ left˙on = 'WWWHH', right˙on = 'WWWHH') df˙growth = pd.merge(df˙growth, oil˙consumption,“ left˙on = 'WWWHH', right˙on = 'WWWHH') df˙growth = pd.merge(df˙growth, vegetables˙consumption,“ left˙on = 'WWWHH', right˙on = 'WWWHH') df˙growth = pd.merge(df˙growth, fruits˙consumption,“ left˙on = 'WWWHH', right˙on = 'WWWHH') df˙growth = pd.merge(df˙growth, meat˙consumption,“ left˙on = 'WWWHH', right˙on = 'WWWHH') df˙growth = pd.merge(df˙growth, sugar˙consumption,“ left˙on = 'WWWHH', right˙on = 'WWWHH') df˙growth['total'] = df˙growth.sum(axis=1) df˙growth = percentage˙share(df˙growth) df˙growth.drop('total',axis=1,inplace=**True**) df˙growth['cluster˙label'] = kmeans.predict(ddf˙scaled) df˙g = df˙growth.set˙index('cluster˙label') g˙share˙cluster = df˙g.groupby('cluster˙label')[df˙g.columns].mean()

[commandchars=
{}] /anaconda3/lib/python3.7/site-packages/ipykernel_launcher.py:25: RuntimeWarning: invalid value encountered in double_scalars

[commandchars=
{}] In [116]: list1˙=[] **for** rows **in** p˙share˙cluster.itertuples(): *# Create list for the current row* my˙list =[rows.grains˙cerials, rows.pulses˙lentals, rows.dairy, rows.oil˙fat, rows.vegetables, rows.fruits, rows.meat˙fish, rows.pulses˙lentals, rows.sugar]

 # append the list to the final list list1˙.append(my˙list)

list2˙=[] **for** rows **in** g˙share˙cluster.itertuples(): *# Create list for the current row* my˙list =[rows.grains˙cerials, rows.pulses˙lentals, rows.dairy, rows.oil˙fat, rows.vegetables, rows.fruits, rows.meat˙fish, rows.pulses˙lentals]

 # append the list to the final list list2˙.append(my˙list)

inc1 = ['grains˙cerials', 'pulses˙lentals', 'poultry˙dairy', 'oil˙fat', 'vegetables', 'fruits', 'meat˙fish', 'sugar'] inc2 = ['grains˙cerials', 'pulses˙lentals', 'poultry˙dairy', 'oil˙fat', 'vegetables', 'fruits', 'meat˙fish', 'pulses˙lentals']

c˙1 = list1˙[1] c˙2 = list1˙[2] c˙3 = list1˙[3]

def list˙avg(l1,l2,l3): ll = [] **for** i **in** range(len(l1)): val = (l1[i]+l2[i]+l3[i])/3 ll.append(val) **return** ll

trad˙home = list˙avg(c˙1,c˙2,c˙3) c˙4 = list1˙[0] c˙5 = list1˙[4]

c˙11 = list2˙[1] c˙22 = list2˙[2] c˙33 = list2˙[3] trad˙home2 = list˙avg(c˙11,c˙22,c˙33) c˙44 = list2˙[0] c˙55 = list2˙[4]

[commandchars=
{}] In [118]: fig = plt.figure() ax1 = plt.subplot(1,2,1)

explode = (0.1,0.1,0.1,0.1,0.1,0.1,0.1,0.1)

def func2(pct, allvals): absolute = int(pct/100.*np.sum(allvals)) **return** "–:.1f"%".format(absolute)

wedges, texts, autotexts = ax1.pie(trad˙home2,explode=explode, autopct=**lambda** pct: func2(pct,trad˙home2), textprops=dict(color="k"))

plt.setp(autotexts, size=12, weight="bold")

def func2(pct, allvals): absolute = int(pct/100.*np.sum(allvals)) return "–:.1f"%".format(absolute)
ax2 = plt.subplot(1,2,2)
explode = (0.1,0.1,0.1,0.1,0.1,0.1,0.1,0.1,0.1)
wedges, texts, autotexts = ax2.pie(trad˙home,explode=explode,autopct=lambda pct: func2(pct,trad˙home), textprops=dict(color="k"))
ax2.legend(wedges,inc1, loc="best", title = 'Food-types', bbox˙to˙anchor=(1, 0, 0.5, 1))
plt.setp(autotexts, size=12, weight="bold")
ax1.set˙title('Percentage share of average monetary value for home-grown food-types',“ y=1,fontsize=15,fontweight='bold') ax2.set˙title('Percentage share of average monetary value for Purchased food-types',“ y=1,fontsize=15,fontweight='bold')
#Set the figure size so that all four panel graphs are clearly visible fig.set˙size˙inches(20,10) *#Set a title for the figure* fig.suptitle('Cluster: Traditional-Home-Producers ',y=1,fontsize=20,fontweight='bold')
plt.show()

max size=0.90.9output$_3$3$_0$*.png*

[commandchars=
{}] In [119]: fig = plt.figure() ax1 = plt.subplot(1,2,1)
explode = (0.1,0.1,0.1,0.1,0.1,0.1,0.1,0.1)
def func2(pct, allvals): absolute = int(pct/100.*np.sum(allvals)) return "–:.1f"%".format(absolute)
wedges, texts, autotexts = ax1.pie(c˙55,explode=explode, autopct=lambda pct: func2(pct,c˙55), textprops=dict(color="k"))
plt.setp(autotexts, size=12, weight="bold")
ax2 = plt.subplot(1,2,2)
explode = (0.1,0.1,0.1,0.1,0.1,0.1,0.1,0.1,0.1)
wedges, texts, autotexts = ax2.pie(c˙5,explode=explode,autopct=lambda pct: func2(pct,c˙5), textprops=dict(color="k")) ax2.legend(wedges,inc1, loc="best", title = 'Food-types', bbox˙to˙anchor=(1, 0, 0.5, 1))
plt.setp(autotexts, size=12, weight="bold")
ax1.set˙title('Percentage share of average monetary value for home-grown food-types '“ ,y=1,fontsize=15,fontweight='bold') ax2.set˙title('Percentage share of average monetary value for Purchased food-types'“ ,y=1,fontsize=15,fontweight='bold')
#Set the figure size so that all four panel graphs are clearly visible fig.set˙size˙inches(20,10) *#Set a title for the figure* fig.suptitle('Cluster: Non-traditional-Home-Producer',y=1,fontsize=20,fontweight='bold') plt.show()

max size=0.90.9output$_3$4$_0$*.png*

[commandchars=
{}] In [121]: fig = plt.figure() ax1 = plt.subplot(1,2,1)
explode = (0.1,0.1,0.1,0.1,0.1,0.1,0.1,0.1)
def func2(pct, allvals): absolute = int(pct/100.*np.sum(allvals)) return "–:.1f"%".format(absolute)

wedges, texts, autotexts = ax1.pie(c˙44,explode=explode, autopct=**lambda** pct: func2(pct,c˙44), textprops=dict(color="k"))

plt.setp(autotexts, size=12, weight="bold")

ax2 = plt.subplot(1,2,2)

explode = (0.1,0.1,0.1,0.1,0.1,0.1,0.1,0.1,0.1)

wedges, texts, autotexts = ax2.pie(c˙4,explode=explode,autopct=**lambda** pct: func2(pct,c˙4), textprops=dict(color="k"))

ax2.legend(wedges,inc1, loc="best", title = 'Food-types', bbox˙to˙anchor=(1, 0, 0.5, 1))

plt.setp(autotexts, size=12, weight="bold")

ax1.set˙title('Percentage share of average monetary value forhome-grown food-types '" ,y=1,fontsize=15,fontweight='bold') ax2.set˙title('Percentage share of average monetary value for Purchased food-types'," y=1,fontsize=15,fontweight='bold')

#Set the figure size so that all four panel graphs are clearly visible fig.set˙size˙inches(20,10) *#Set a title for the figure* fig.suptitle('Non-traditional-Purchaser',y=1,fontsize=20,fontweight='bold') plt.show()

max size=0.90.9output$_3$5$_0$.*png*

Basket Anslysis

[commandchars=

{}] In [128]: df˙association = pd.DataFrame(food˙nlss1[food˙nlss1.item ¿= 11]" [food˙nlss1[food˙nlss1.item ¿= 11].item ¡= 94]" .set˙index(['WWWHH']))

#Basket for item Purchase

purchase˙basket = df˙association.groupby(['WWWHH','item'])['amount˙purchased']" .sum() " .unstack() " .reset˙index() " .fillna(0) " .set˙index('WWWHH')

 # recode all multiple purchases to 1 **def** encode˙units(x): **if** x ¡= 0: **return** 0 **if** x ¿= 1: **return** 1 purchase˙basket˙sets = purchase˙basket.applymap(encode˙units)

 # remove all household purchasing less than 5 different items

purchase˙basket˙sets = purchase˙basket˙sets[purchase˙basket˙sets.sum(axis = 1) ¿ 15]" .fillna(0.00) purchase˙basket˙sets.columns=['Fine rice','Coarse rice','Beaten/Flattened rice' ," 'Maize','Maize flour','Wheat flour','Millet'," 'Other grains/cerials','Black pulse','Masoor'," 'Rahar','Gram','Other pulses','Other beans','Eggs'" ,'Milk','Condensed milk','Baby milk/powder milk'," 'Curd','Other milk Products','Ghee','vegetable oil'," 'Mustard oil','Other oil','Potatoes/Sweet Potatoes'," 'Onions','Cauliflower/Cabbage','Tomatoes'," 'Greeen leafy Vegetables','Other Vegetables','Bananas'," 'Citrus Fruits','Mangoes','Apples','Pineapple','Papaya'," 'Other fruits','Dried fruits','Fish','Mutton','Buffalo meat'," 'Chicken','Other meats','Salt','Cumin seed/black pepper'," 'turmeric','Ginger and Garlic','chilies'," 'Other spices and condements','Sugar'," 'Caramel','Sweets','Sugar Candy'] purchase˙basket˙sets.drop(['Salt','Cumin seed/black pepper','turmeric','Ginger and Garlic', 'chilies','Other spices and condements','Sugar'," 'Caramel','Sweets','Sugar Candy'],axis=1,inplace=**True**)

#Basket for item grown

growth˙basket = df˙association.groupby(['WWWHH','item'])['amount˙grown']" .sum() " .unstack() " .reset˙index() " .fillna(0) " .set˙index('WWWHH')

 # recode all multiple purchases to 1 **def** encode˙units(x): **if** x ¡= 0: **return** 0 **if** x ¿= 1: **return** 1 growth˙basket˙sets = growth˙basket.applymap(encode˙units)

 # remove all household purchasing less than 5 different items

growth˙basket˙sets = growth˙basket˙sets[growth˙basket˙sets.sum(axis = 1) ¿ 15]“ .fillna(0.00) growth˙basket˙sets.columns=['Fine rice','Coarse rice','Beaten/Flattened rice' ,“ 'Maize','Maize flour','Wheat flour','Millet',“ 'Other grains/cerials','Black pulse','Masoor',“ 'Rahar','Gram','Other pulses','Other beans','Eggs'“ ,'Milk','Condensed milk','Baby milk/powder milk',“ 'Curd','Other milk Products','Ghee','vegetable oil',“ 'Mustard oil','Other oil','Potatoes/Sweet Potatoes',“ 'Onions','Cauliflower/Cabbage','Tomatoes',“ 'Greeen leafy Vegetables','Other Vegetables','Bananas',“ 'Citrus Fruits','Mangoes','Apples','Pineapple','Papaya',“ 'Other fruits','Dried fruits','Fish','Mutton','Buffalo meat',“ 'Chicken','Other meats','Salt','Cumin seed/black pepper',“ 'turmeric','Ginger and Garlic','chilies',“ 'Other spices and condements','Sugar',“ 'Caramel','Sweets','Sugar Candy'] growth˙basket˙sets.drop(['Salt','Cumin seed/black pepper','turmeric','Ginger and Garlic', 'chilies','Other spices and condements','Sugar',“ 'Caramel','Sweets','Sugar Candy'],axis=1,inplace=True)

[commandchars=
{}] In [129]: purchase˙basket˙sets.head()

[commandchars=
{}] Out[129]: Fine rice Coarse rice Beaten/Flattened rice Maize Maize flour \ WWWHH 301.0 1 0.0 1.0 0.0 0 302.0 1 0.0 1.0 0.0 0 305.0 1 0.0 1.0 0.0 0 306.0 1 0.0 1.0 0.0 0 307.0 1 0.0 1.0 0.0 0

Wheat flour Millet Other grains/cerials Black pulse Masoor ... \ WWWHH ... 301.0 0.0 0 0 1.0 1.0 ... 302.0 0.0 0 0 0.0 0.0 ... 305.0 1.0 0 0 1.0 1.0 ... 306.0 1.0 0 0 1.0 1.0 ... 307.0 1.0 0 0 1.0 1.0 ...

Apples Pineapple Papaya Other fruits Dried fruits Fish Mutton \ WWWHH 301.0 1.0 1.0 1 0 0 1.0 1.0 302.0 1.0 0.0 0 0 0 1.0 0.0 305.0 1.0 0.0 0 0 0 0.0 1.0 306.0 1.0 0.0 1 0 0 1.0 1.0 307.0 1.0 1.0 0 0 0 1.0 1.0

Buffalo meat Chicken Other meats WWWHH 301.0 1.0 1.0 0 302.0 1.0 0.0 0 305.0 0.0 1.0 0 306.0 0.0 1.0 0 307.0 0.0 1.0 0

[5 rows x 43 columns]

[commandchars=
{}] In [130]: growth˙basket˙sets.head()

[commandchars=
{}] Out[130]: Fine rice Coarse rice Beaten/Flattened rice Maize Maize flour \ WWWHH 105.0 0 1 0 1.0 1 107.0 1 1 0 1.0 0 114.0 0 1 1 1.0 1 116.0 1 1 1 1.0 0 605.0 0 1 0 1.0 1

Wheat flour Millet Other grains/cerials Black pulse Masoor ... \ WWWHH ... 105.0 1.0 0 0 1.0 1.0 ... 107.0 1.0 0 0 1.0 1.0 ... 114.0 1.0 0 0 0.0 1.0 ... 116.0 1.0 0 0 1.0 1.0 ... 605.0 1.0 1 0 1.0 0.0 ...

Apples Pineapple Papaya Other fruits Dried fruits Fish Mutton \ WWWHH 105.0 0 0 1 0 0 1.0 0.0 107.0 0 0 1 0 0 0.0 0.0 114.0 1 0 0 0 0 0.0 1.0 116.0 0 0 1 0 0 0.0 0.0 605.0 0 0 1 0 0 0.0 0.0

Buffalo meat Chicken Other meats WWWHH 105.0 0.0 1.0 0 107.0 0.0 0.0 0 114.0 0.0 1.0 0 116.0 0.0 0.0 0 605.0 0.0 0.0 0

[5 rows x 43 columns]

[commandchars=
{}] In [172]: trad˙hh = df˙expence[df˙expence['cluster˙label'] ¡=3] trad˙hh = trad˙hh[trad˙hh['cluster˙label'] ¿=0]['cluster˙label'] trad˙hh˙purchase˙basket = pd.merge(trad˙hh, purchase˙basket˙sets,“ left˙on = 'WWWHH', right˙on = 'WWWHH') trad˙hh˙purchase˙basket.drop('cluster˙label',axis=1,inplace=True) frequent˙itemsets = apriori(trad˙hh˙purchase˙basket, min˙support=0.5,“ use˙colnames=True, verbose=0, low˙memory=True) rules = association˙rules(frequent˙itemsets, metric='confidence',

min˙threshold=0.5)

df˙results1 = pd.DataFrame(rules) df˙results1.shape df˙results1.drop('conviction',axis=1,inplace=**True**) df˙results1=df˙results1[df˙results1["lift"] ¿ 1.2] df˙results1.sort˙values(["confidence"],ascending=**False**)

[commandchars=
{}] Out[172]: antecedents \ 13023 (Potatoes/Sweet Potatoes, Onions, Apples) 6906 (Potatoes/Sweet Potatoes, Apples) 6778 (Potatoes/Sweet Potatoes, Apples) 13033 (Potatoes/Sweet Potatoes, Apples) 7201 (Onions, Apples) 13037 (Onions, Apples) 17295 (Bananas, Milk, Cauliflower/Cabbage) 7205 (Apples) 7192 (Citrus Fruits, Tomatoes, Onions) 17234 (Mangoes, Potatoes/Sweet Potatoes, Tomatoes, O... 13028 (Potatoes/Sweet Potatoes, Citrus Fruits, Cauli... 13032 (Citrus Fruits, Onions, Cauliflower/Cabbage) 6779 (Citrus Fruits, Tomatoes) 7196 (Citrus Fruits, Tomatoes) 6903 (Mangoes, Citrus Fruits) 13042 (Citrus Fruits, Cauliflower/Cabbage)

consequents antecedent support \ 13023 (Citrus Fruits, Cauliflower/Cabbage) 0.569959 6906 (Mangoes, Citrus Fruits) 0.592593 6778 (Citrus Fruits, Tomatoes) 0.592593 13033 (Citrus Fruits, Onions, Cauliflower/Cabbage) 0.592593 7201 (Citrus Fruits, Tomatoes) 0.593621 13037 (Potatoes/Sweet Potatoes, Citrus Fruits, Cauli... 0.593621 17295 (Mangoes, Potatoes/Sweet Potatoes, Tomatoes, O... 0.613169 7205 (Citrus Fruits, Tomatoes, Onions) 0.627572 7192 (Apples) 0.663580 17234 (Bananas, Milk, Cauliflower/Cabbage) 0.687243 13028 (Onions, Apples) 0.694444 13032 (Potatoes/Sweet Potatoes, Apples) 0.695473 6779 (Potatoes/Sweet Potatoes, Apples) 0.701646 7196 (Onions, Apples) 0.701646 6903 (Potatoes/Sweet Potatoes, Apples) 0.710905 13042 (Potatoes/Sweet Potatoes, Onions, Apples) 0.728395

consequent support support confidence lift leverage 13023 0.728395 0.500000 0.877256 1.204369 0.084845 6906 0.710905 0.506173 0.854167 1.201520 0.084896 6778 0.701646 0.502058 0.847222 1.207478 0.086267 13033 0.695473 0.500000 0.843750 1.213203 0.087868 7201 0.701646 0.500000 0.842288 1.200445 0.083488 13037 0.694444 0.500000 0.842288 1.212894 0.087763 17295 0.687243 0.508230 0.828859 1.206064 0.086835 7205 0.663580 0.500000 0.796721 1.200640 0.083556 7192 0.627572 0.500000 0.753488 1.200640 0.083556 17234 0.613169 0.508230 0.739521 1.206064 0.086835 13028 0.593621 0.500000 0.720000 1.212894 0.087763 13032 0.592593 0.500000 0.718935 1.213203 0.087868 6779 0.592593 0.502058 0.715543 1.207478 0.086267 7196 0.593621 0.500000 0.712610 1.200445 0.083488 6903 0.592593 0.506173 0.712012 1.201520 0.084896 13042 0.569959 0.500000 0.686441 1.204369 0.084845

[commandchars=
{}] In [171]: trad˙hh = df˙expence[df˙expence['cluster˙label'] ¡=3] trad˙hh = trad˙hh[trad˙hh['cluster˙label'] ¿=0]['cluster˙label'] trad˙hh˙growth˙basket = pd.merge(trad˙hh, growth˙basket˙sets,“ left˙on = 'WWWHH', right˙on = 'WWWHH') trad˙hh˙growth˙basket.drop('cluster˙label',axis=1,inplace=**True**) frequent˙itemsets = apriori(trad˙hh˙growth˙basket, min˙support=0.5,“ use˙colnames=**True**, verbose=0, low˙memory=**True**) rules = association˙rules(frequent˙itemsets, metric='confidence', min˙threshold=0.1)

df˙results1 = pd.DataFrame(rules) df˙results1.shape df˙results1.drop('conviction',axis=1,inplace=**True**) df˙results1 = df˙results1[df˙results1["lift"] ¿ 1.166] df˙results1.sort˙values(["confidence"],ascending=**False**)

[commandchars=
{}] Out[171]: antecedents \ 12273 (Maize, Ghee, Cauliflower/Cabbage) 19784 (Maize, Potatoes/Sweet Potatoes, Ghee, Caulifl... 15731 (Bananas, Tomatoes, Onions) 22741 (Bananas, Ghee, Tomatoes) 7272 (Cauliflower/Cabbage, Mustard oil) 15637 (Mustard oil, Potatoes/Sweet

Potatoes, Caulifl... 5116 (Maize, Cauliflower/Cabbage) 13114 (Maize, Potatoes/Sweet Potatoes, Cauliflower/C... 15651 (Mustard oil, Cauliflower/Cabbage) 5117 (Tomatoes, Onions) 7273 (Tomatoes, Onions) 15634 (Potatoes/Sweet Potatoes, Tomatoes, Onions) 13131 (Tomatoes, Onions) 15648 (Tomatoes, Onions) 22688 (Maize, Potatoes/Sweet Potatoes, Milk, Mustard... 15734 (Potatoes/Sweet Potatoes, Cauliflower/Cabbage) 12292 (Tomatoes, Milk) 19833 (Tomatoes, Milk)

consequents antecedent support \ 12273 (Tomatoes, Milk) 0.540816 19784 (Tomatoes, Milk) 0.530612 15731 (Potatoes/Sweet Potatoes, Cauliflower/Cabbage) 0.591837 22741 (Maize, Potatoes/Sweet Potatoes, Milk, Mustard... 0.622449 7272 (Tomatoes, Onions) 0.642857 15637 (Tomatoes, Onions) 0.632653 5116 (Tomatoes, Onions) 0.653061 13114 (Tomatoes, Onions) 0.642857 15651 (Potatoes/Sweet Potatoes, Tomatoes, Onions) 0.642857 5117 (Maize, Cauliflower/Cabbage) 0.693878 7273 (Cauliflower/Cabbage, Mustard oil) 0.693878 15634 (Mustard oil, Cauliflower/Cabbage) 0.683673 13131 (Maize, Potatoes/Sweet Potatoes, Cauliflower/C... 0.693878 15648 (Mustard oil, Potatoes/Sweet Potatoes, Caulifl... 0.693878 22688 (Bananas, Ghee, Tomatoes) 0.714286 15734 (Bananas, Tomatoes, Onions) 0.724490 12292 (Maize, Ghee, Cauliflower/Cabbage) 0.806122 19833 (Maize, Potatoes/Sweet Potatoes, Ghee, Caulifl... 0.806122

consequent support support confidence lift leverage 12273 0.806122 0.510204 0.943396 1.170289 0.074240 19784 0.806122 0.500000 0.942308 1.168939 0.072262 15731 0.724490 0.500000 0.844828 1.166100 0.071220 22741 0.714286 0.520408 0.836066 1.170492 0.075802 7272 0.693878 0.530612 0.825397 1.189542 0.084548 15637 0.693878 0.520408 0.822581 1.185484 0.081424 5116 0.693878 0.530612 0.812500 1.170956 0.077468 13114 0.693878 0.520408 0.809524 1.166667 0.074344 15651 0.683673 0.520408 0.809524 1.184080 0.080904 5117 0.653061 0.530612 0.764706 1.170956 0.077468 7273 0.642857 0.530612 0.764706 1.189542 0.084548 15634 0.642857 0.520408 0.761194 1.184080 0.080904 13131 0.642857 0.520408 0.750000 1.166667 0.074344 15648 0.632653 0.520408 0.750000 1.185484 0.081424 22688 0.622449 0.520408 0.728571 1.170492 0.075802 15734 0.591837 0.500000 0.690141 1.166100 0.071220 12292 0.540816 0.510204 0.632911 1.170289 0.074240 19833 0.530612 0.500000 0.620253 1.168939 0.072262

[commandchars=
{}] In [149]: trad˙purc = df˙expence[df˙expence['cluster˙label'] == 0]['cluster˙label'] trad˙purc˙purchase˙basket = pd.merge(trad˙purc, purchase˙basket˙sets,“ left˙on = 'WWWHH', right˙on = 'WWWHH') trad˙purc˙purchase˙basket.drop('cluster˙label',axis=1,inplace=**True**) frequent˙itemsets = apriori(trad˙purc˙purchase˙basket, min˙support=0.5,“ use˙colnames=**True**, verbose=0, low˙memory=**True**) rules = association˙rules(frequent˙itemsets, metric='confidence', min˙threshold=0.5)

df˙results1 = pd.DataFrame(rules) df˙results1.shape df˙results1.drop('conviction',axis=1,inplace=**True**) df˙results1 = df˙results1[df˙results1["lift"] ¿ 1.2] df˙results1.sort˙values(["confidence"],ascending=**False**)

[commandchars=
{}] Out[149]: antecedents \ 15827 (Onions, Apples) 15823 (Potatoes/Sweet Potatoes, Apples) 15943 (Potatoes/Sweet Potatoes, Apples) 15707 (Onions, Apples) 15703 (Potatoes/Sweet Potatoes, Apples) 15822 (Bananas, Citrus Fruits, Onions) 15702 (Citrus Fruits, Tomatoes, Onions) 15818 (Bananas, Potatoes/Sweet Potatoes, Citrus Fruits) 15942 (Mangoes, Citrus Fruits, Onions) 15698 (Potatoes/Sweet Potatoes, Citrus Fruits, Tomat...

consequents antecedent support \ 15827 (Bananas, Potatoes/Sweet Potatoes, Citrus Fruits) 0.608324 15823 (Bananas, Citrus Fruits, Onions) 0.610459 15943 (Mangoes, Citrus Fruits, Onions) 0.610459 15707 (Potatoes/Sweet Potatoes, Citrus Fruits, Tomat... 0.608324 15703 (Citrus Fruits, Tomatoes, Onions) 0.610459 15822 (Potatoes/Sweet Potatoes, Apples) 0.686233 15702 (Potatoes/Sweet Potatoes, Apples) 0.677695 15818 (Onions, Apples) 0.688367 15942 (Potatoes/Sweet Potatoes, Apples) 0.684098 15698 (Onions, Apples) 0.685165

consequent support support confidence lift leverage 15827 0.688367 0.506937 0.833333 1.210594 0.088186 15823 0.686233 0.506937 0.830420 1.210114 0.088020 15943 0.684098 0.502668 0.823427 1.203667 0.085054 15707 0.685165 0.500534 0.822807 1.200888 0.083731 15703 0.677695 0.500534 0.819930 1.209881 0.086829 15822 0.610459 0.506937 0.738725 1.210114 0.088020 15702 0.610459 0.500534 0.738583 1.209881 0.086829 15818 0.608324 0.506937 0.736434 1.210594 0.088186 15942 0.610459 0.502668 0.734789 1.203667 0.085054 15698 0.608324 0.500534 0.730530 1.200888 0.083731

[commandchars=
{}] In [162]: trad˙purc = df˙expence[df˙expence['cluster˙label'] == 0]['cluster˙label'] trad˙purc˙growth˙basket = pd.merge(trad˙purc, growth˙basket˙sets,“ left˙on = 'WWWHH', right˙on = 'WWWHH') trad˙purc˙growth˙basket.drop('cluster˙label',axis=1,inplace=**True**) frequent˙itemsets = apriori(trad˙purc˙growth˙basket, min˙support=0.5,“ use˙colnames=**True**, verbose=0, low˙memory=**True**) rules = association˙rules(frequent˙itemsets, metric='confidence', min˙threshold=0.5)

df˙results1 = pd.DataFrame(rules) df˙results1.shape df˙results1.drop('conviction',axis=1,inplace=**True**) df˙results1 = df˙results1[df˙results1["lift"] ¿ 1.0] df˙results1.sort˙values(["confidence"],ascending=**False**)

[commandchars=
{}] Out[162]: antecedents \ 24385 (Potatoes/Sweet Potatoes, Maize, Ghee, Wheat f... 13471 (Maize, Wheat flour, Ghee, Tomatoes) 13531 (Maize, Wheat flour, Ghee, Papaya) 25872 (Potatoes/Sweet Potatoes, Maize, Onions, Ghee,... 25828 (Maize, Bananas, Ghee, Mustard oil) ... ... 30029 (Potatoes/Sweet Potatoes) 20575 (Potatoes/Sweet Potatoes) 9223 (Potatoes/Sweet Potatoes) 2190 (Potatoes/Sweet Potatoes) 3274 (Potatoes/Sweet Potatoes)

consequents antecedent support \ 24385 (Milk) 0.5625 13471 (Milk) 0.6125 13531 (Milk) 0.5125 25872 (Milk) 0.5000 25828 (Potatoes/Sweet Potatoes, Milk) 0.5500 ... ... 30029 (Maize, Ghee, Milk, Wheat flour, Bananas, Must... 0.9875 20575 (Mustard oil, Bananas, Ghee, Cauliflower/Cabbage) 0.9875 9223 (Bananas, Tomatoes, Papaya) 0.9875 2190 (Mangoes, Onions) 0.9875 3274 (Mangoes, Wheat flour, Coarse rice) 0.9875

consequent support support confidence lift leverage 24385 0.9250 0.5625 1.000000 1.081081 0.042187 13471 0.9250 0.6125 1.000000 1.081081 0.045937 13531 0.9250 0.5125 1.000000 1.081081 0.038437 25872 0.9250 0.5000 1.000000 1.081081 0.037500 25828 0.9125 0.5500 1.000000 1.095890 0.048125 ... ... 30029 0.5000 0.5000 0.506329 1.012658 0.006250 20575 0.5000 0.5000 0.506329 1.012658 0.006250 9223 0.5000 0.5000 0.506329 1.012658 0.006250 2190 0.5000 0.5000 0.506329 1.012658 0.006250 3274 0.5000 0.5000 0.506329 1.012658 0.006250

[18167 rows x 8 columns]

[commandchars=
{}] In [165]: trad˙home = df˙expence[df˙expence['cluster˙label'] == 4]['cluster˙label'] trad˙home˙purchase˙basket = pd.merge(trad˙home, purchase˙basket˙sets,“ left˙on = 'WWWHH', right˙on = 'WWWHH') trad˙home˙purchase˙basket.drop('cluster˙label',axis=1,inplace=**True**) frequent˙itemsets = apriori(trad˙home˙purchase˙basket, min˙support=0.5,“ use˙colnames=**True**, verbose=0, low˙memory=**True**) rules = association˙rules(frequent˙itemsets, metric='confidence', min˙threshold=0.5)

df˙results1 = pd.DataFrame(rules) df˙results1.shape df˙results1.drop('conviction',axis=1,inplace=**True**) df˙results1 = df˙results1[df˙results1["lift"] ¿ 1.15] df˙results1.sort˙values(["confidence"],ascending=**False**)

[commandchars=
{}] Out[165]: antecedents \ 16 (Wheat flour) 10314 (Mangoes, Tomatoes, Milk, Apples) 10137 (Mangoes, Citrus Fruits, Tomatoes) 4229 (Curd, Potatoes/Sweet Potatoes, Onions) 5602 (Potatoes/Sweet Potatoes, Apples, Mutton) ... ... 1831 (Mustard oil) 11989 (Citrus Fruits) 1839 (Milk, Mustard oil) 11929 (Citrus Fruits) 14623 (Citrus Fruits)

consequents antecedent support \ 16 (Mustard oil) 0.529412 10314 (Citrus Fruits) 0.529412 10137 (Bananas, Milk) 0.588235 4229 (Citrus Fruits) 0.529412 5602 (Citrus Fruits) 0.588235 ... ... 1831 (Beaten/Flattened rice, Milk, Mutton) 0.823529 11989 (Bananas, Tomatoes, Apples, Mutton) 0.823529 1839 (Wheat flour, Mutton) 0.823529 11929 (Mangoes, Bananas, Tomatoes, Apples) 0.823529 14623 (Potatoes/Sweet Potatoes, Mutton, Cauliflower/... 0.823529

consequent support support confidence lift leverage 16 0.823529 0.529412 1.000000 1.214286 0.093426 10314 0.823529 0.529412 1.000000 1.214286 0.093426 10137 0.823529 0.588235 1.000000 1.214286 0.103806 4229 0.823529 0.529412 1.000000 1.214286 0.093426 5602 0.823529 0.588235 1.000000 1.214286 0.103806 ... ... 1831 0.529412 0.529412 0.642857 1.214286 0.093426 11989 0.529412 0.529412 0.642857 1.214286 0.093426 1839 0.529412 0.529412 0.642857 1.214286 0.093426 11929 0.529412 0.529412 0.642857 1.214286 0.093426 14623 0.529412 0.529412 0.642857 1.214286 0.093426

[4716 rows x 8 columns]

[commandchars=
{}] In [94]: trad˙home = df˙expence[df˙expence['cluster˙label'] == 4]['cluster˙label'] trad˙home˙growth˙basket = pd.merge(trad˙home, growth˙basket˙sets,“ left˙on = 'WWWHH', right˙on = 'WWWHH') trad˙home˙growth˙basket.drop('cluster˙label',axis=1,inplace=**True**) frequent˙itemsets = apriori(trad˙home˙growth˙basket, min˙support=0.5,“ use˙colnames=**True**, verbose=0, low˙memory=**True**) rules = association˙rules(frequent˙itemsets, metric='confidence', min˙threshold=0.5)

df˙results1 = pd.DataFrame(rules) df˙results1.shape df˙results1.drop('conviction',axis=1,inplace=**True**) df˙results1 = df˙results1[df˙results1["lift"] ¿ 1.0] df˙results1.sort˙values(["confidence"],ascending=**False**)

[commandchars=
{}] Out[94]: antecedents consequents \ 38 (Coarse rice, Milk) (Wheat flour) 62 (Potatoes/Sweet Potatoes, Milk) (Wheat flour) 17 (Milk) (Wheat flour) 15 (Rahar) (Wheat flour) 67 (Mangoes, Milk) (Wheat flour) .. .. 46 (Wheat flour) (Mangoes, Coarse rice) 18 (Wheat flour) (Mustard oil) 14 (Wheat flour) (Rahar) 11 (Wheat flour) (Beaten/Flattened rice) 39 (Wheat flour) (Coarse rice, Milk)

antecedent support consequent support support confidence lift \ 38 0.500000 0.888889 0.500000 1.0000 1.125000 62 0.583333 0.888889 0.583333 1.0000 1.125000 17 0.722222 0.888889 0.722222 1.0000 1.125000 15 0.500000 0.888889 0.500000 1.0000 1.125000 67 0.583333 0.888889 0.583333 1.0000 1.125000 .. .. 46 0.888889 0.611111 0.555556 0.6250 1.022727 18 0.888889 0.555556 0.500000 0.5625 1.012500 14 0.888889 0.500000 0.500000 0.5625 1.125000 11 0.888889 0.527778 0.500000 0.5625 1.065789 39 0.888889 0.500000 0.500000 0.5625 1.125000

leverage 38 0.055556 62 0.064815 17 0.080247 15 0.055556 67 0.064815 .. ... 46 0.012346 18 0.006173 14 0.055556 11 0.030864 39 0.055556

[70 rows x 8 columns]

[commandchars=
{}] In []: