Association rule mining and Clustering Analysis of Recipes

DATASET

	id	cuisine	ingredients
0	10259	greek	[romaine lettuce, black olives, grape tomatoes...
1	25693	southern_us	[plain flour, ground pepper, salt, tomatoes, g...
2	20130	filipino	[eggs, pepper, salt, mayonaise, cooking oil, g...
3	22213	indian	[water, vegetable oil, wheat, salt]
4	13162	indian	[black pepper, shallots, cornflour, cayenne pe...
...	...	...	...
39769	29109	irish	[light brown sugar, granulated sugar, butter, ...
39770	11462	italian	[KRAFT Zesty Italian Dressing, purple onion, b...
39771	2238	irish	[eggs, citrus fruit, raisins, sourdough starte...
39772	41882	chinese	[boneless chicken skinless thigh, minced garli...
39773	2362	mexican	[green chile, jalapeno chilies, onions, ground...

39774 rows × 3 columns

Association Rule Mining

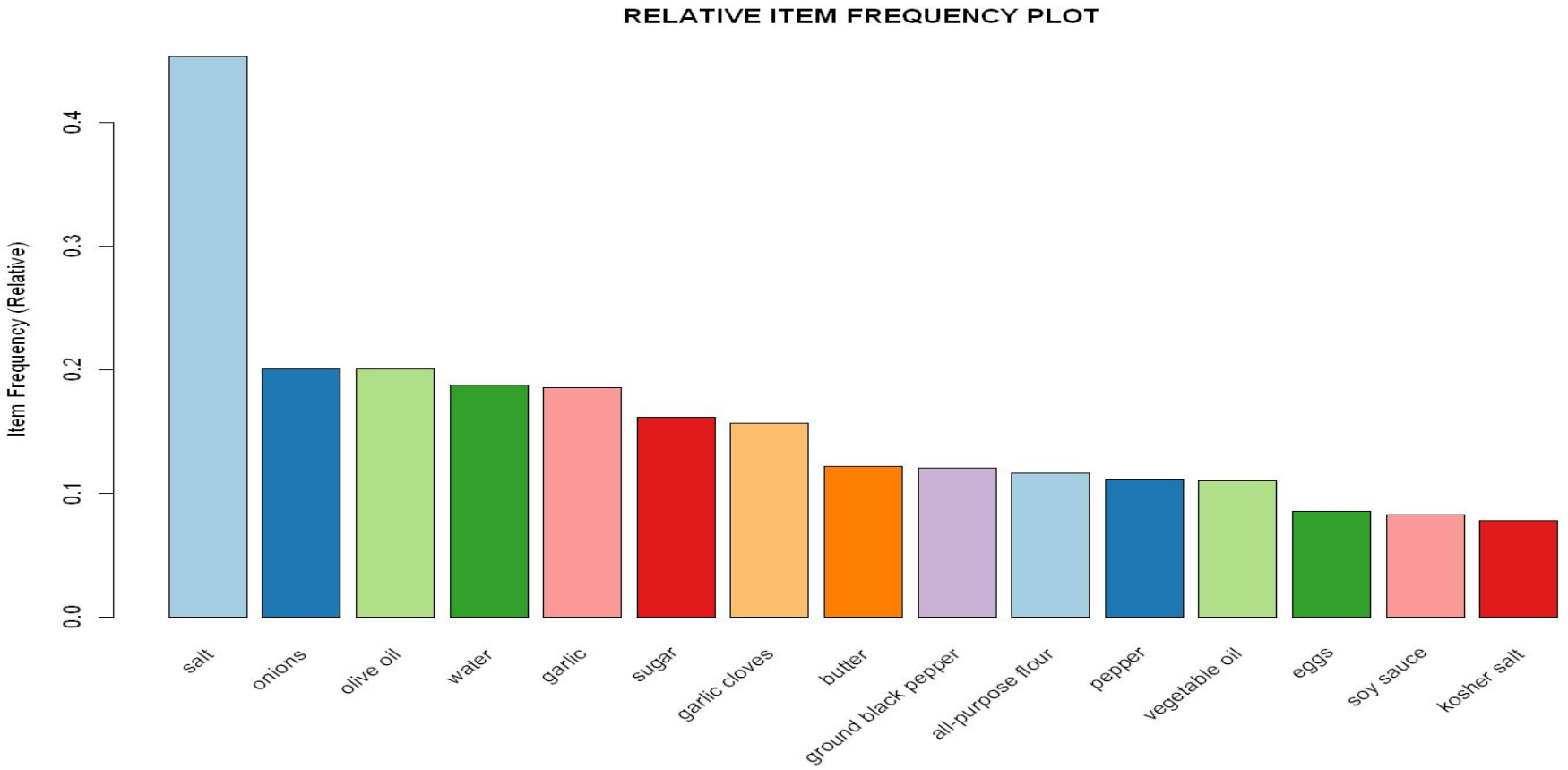
- **Association rule mining is used to find interesting associations and relationships among large sets of data items. This rule shows how frequently an itemset occurs in a transaction.**

- **The Recipes are the transactions and the ingredients in each recipe serve as items.**

- **We want to analyse what ingredients tend to occur together most frequently in recipes.**

- **We will also analyse which ingredients are most likely to occur given a particular set of ingredients.**

RELATIVE ITEM FREQUENCY PLOT



ANALYSIS

The above shown Relative Item frequency plot tells how many times an ingredient has occurred among all transactions as compared to other ingredients. The top 15 ingredients are shown in this plot.

The relative frequency plot shows that Salt constitutes the highest frequency among all recipes. It means that it is used in many recipes.

The relative frequency of salt to other ingredients is quite observable. The Second ingredient is onions which is also widely used among all recipes but not as much as salt.

It means that many recipes use these ingredients.

We used Apriori algorithm for association rule mining.

Experimentations performed

The following table represents the experimentation values used for determining the Association rules and frequent itemsets

```
experimentation_table <- matrix(c(0.5,0.9,0.3,0.9,0.1,0.9,0.008,0.9,0.008,0.6,0.008,0.5),ncol=2,byrow=TRUE)
colnames(experimentation_table) <- c("Min support","Min confidence")
experimentation_table <- as.table(experimentation_table)
experimentation_table
```

```
  Min support Min confidence
A       0.500          0.900
B       0.300          0.900
C       0.100          0.900
D       0.008          0.900
E       0.008          0.600
F       0.008          0.500
```

1. **Association rules with support=0.5 & confidence=0.9**

   We found that no rules exist with these parameters

2. **Association rules with support=0.3 & confidence=0.9**

   We found that no rules exist with these parameters

3. **Association rules with support=0.1 & confidence=0.9**

   We found that no rules exist with these parameters

4. Association rules with support=0.008 & confidence=0.9

We found 8 rules!

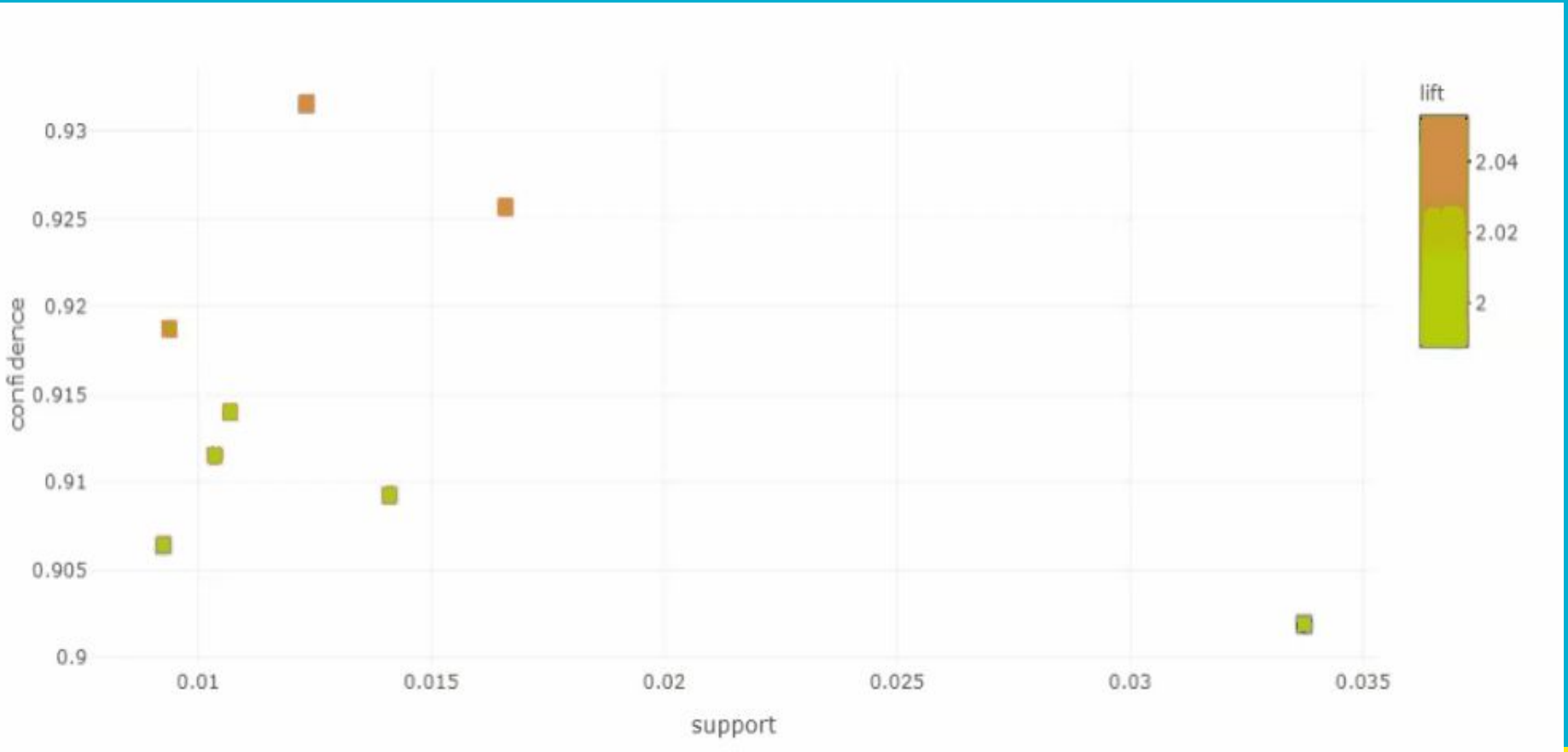
These rules are sorted by lift measure

```
rules.sorted <- sort(rules, by="lift")
```

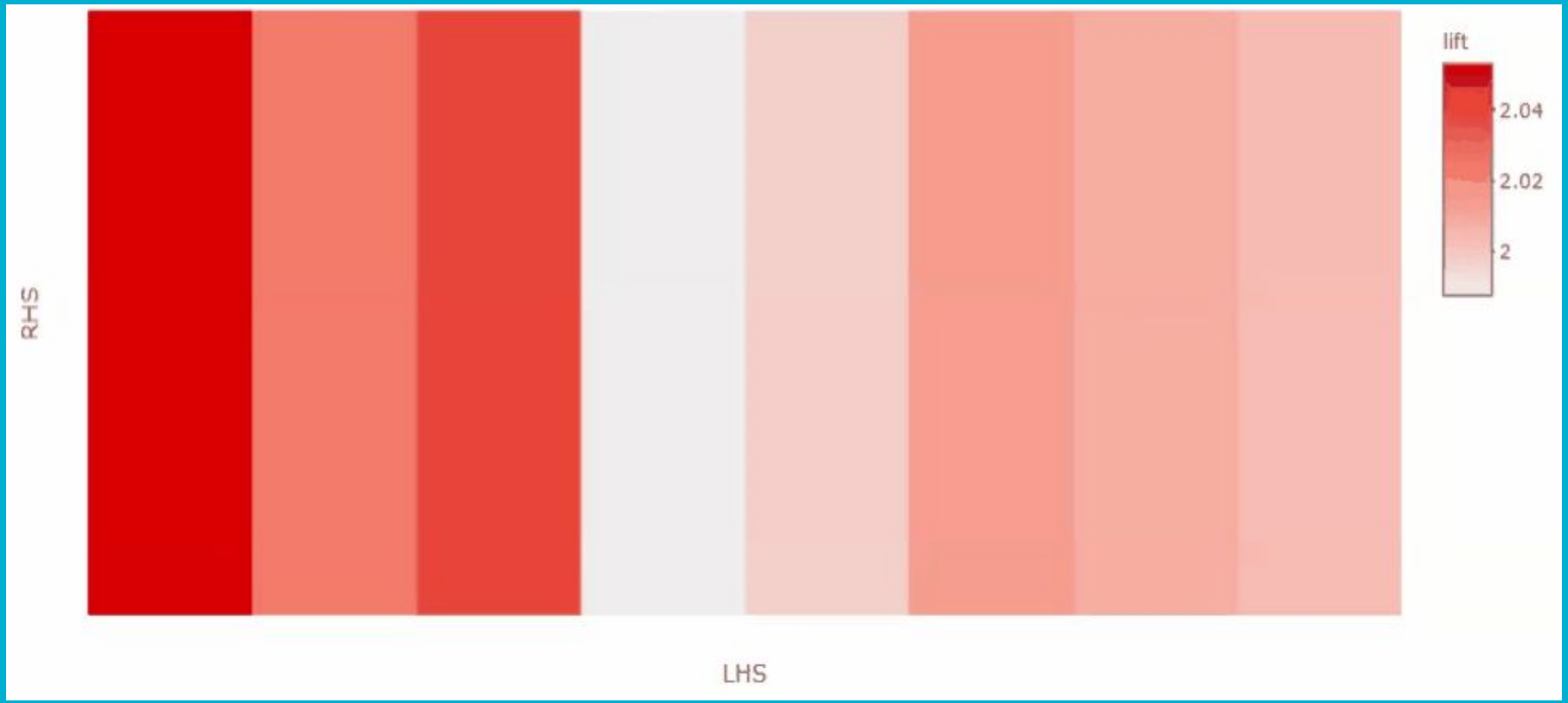
```
inspect(rules.sorted)
```

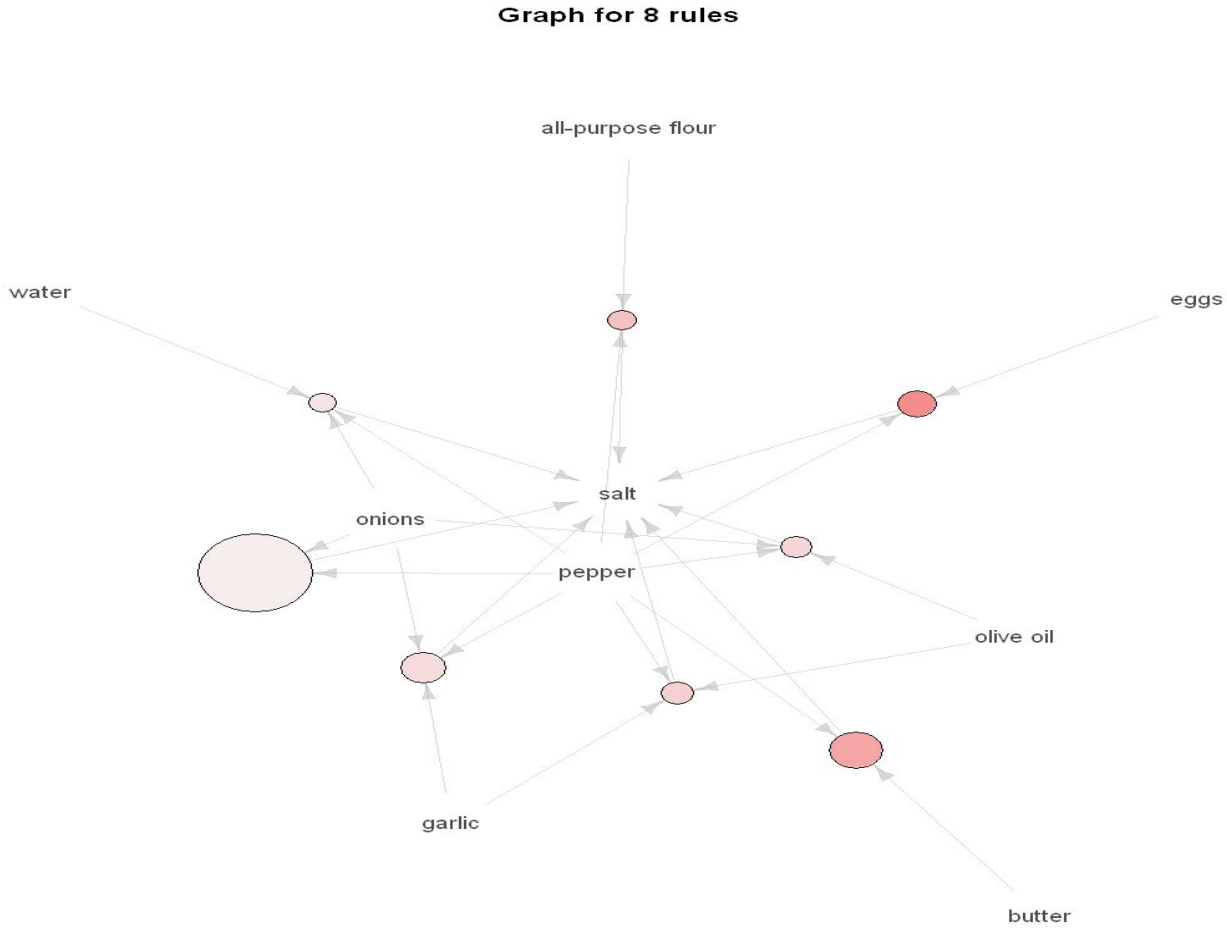
```
     lhs                                rhs     support     confidence coverage
[1] {eggs,pepper}                    => {salt} 0.012319606 0.9315589  0.01322472
[2] {butter,pepper}                  => {salt} 0.016593755 0.9256662  0.01792628
[3] {all-purpose flour,pepper}       => {salt} 0.009377986 0.9187192  0.01020767
[4] {garlic,olive oil,pepper}        => {salt} 0.010685372 0.9139785  0.01169105
[5] {olive oil,onions,pepper}        => {salt} 0.010358526 0.9115044  0.01136421
[6] {garlic,onions,pepper}           => {salt} 0.014104692 0.9092382  0.01551265
[7] {onions,pepper,water}            => {salt} 0.009252275 0.9064039  0.01020767
[8] {onions,pepper}                  => {salt} 0.033740635 0.9018817  0.03741137
     lift     count
[1] 2.052960  490
[2] 2.039974  660
[3] 2.024664  373
[4] 2.014217  425
[5] 2.008764  412
[6] 2.003770  561
[7] 1.997524  368
[8] 1.987558 1342
```

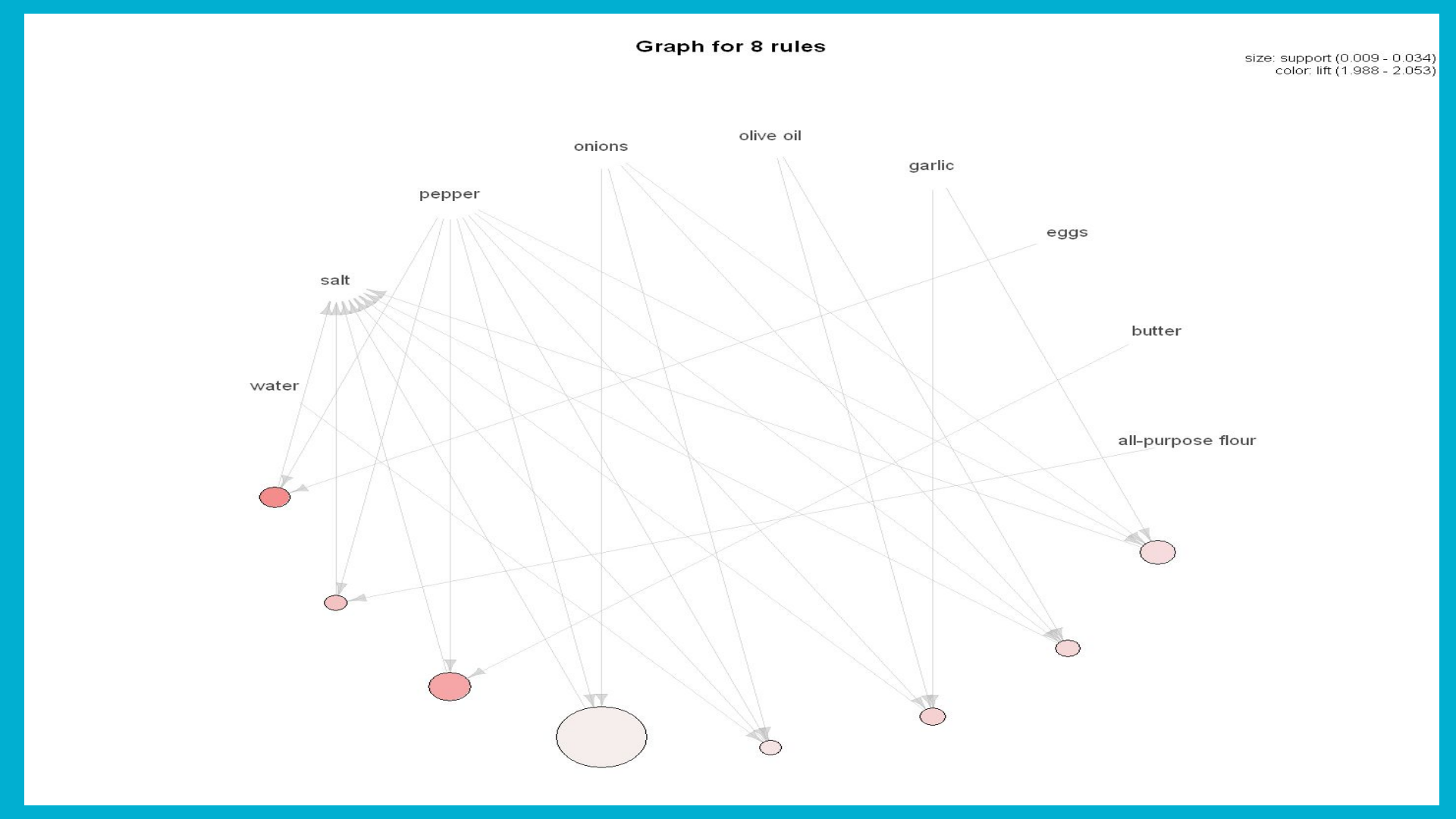
Scatter plot for 8 rules

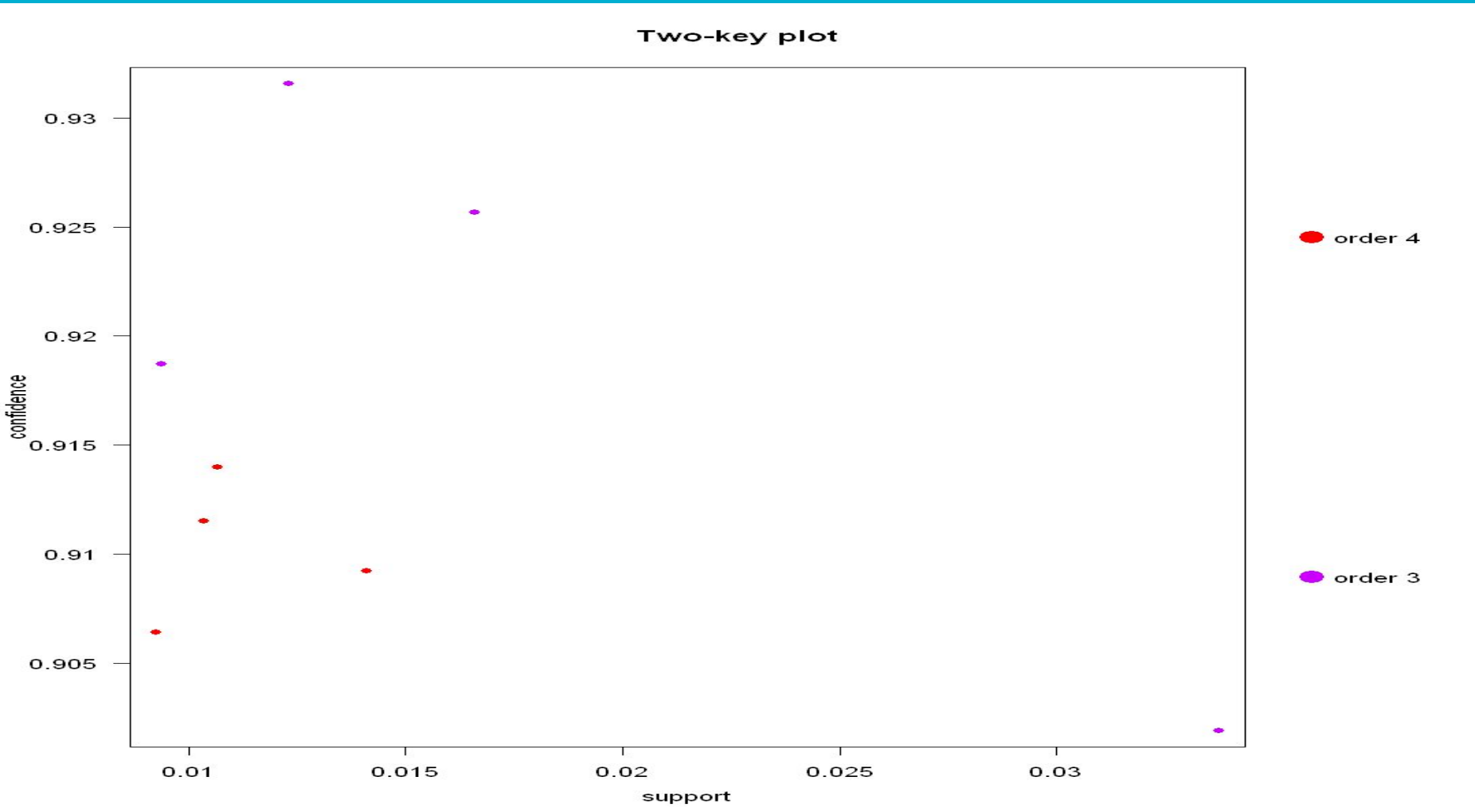


Matrix plot for 8 rules



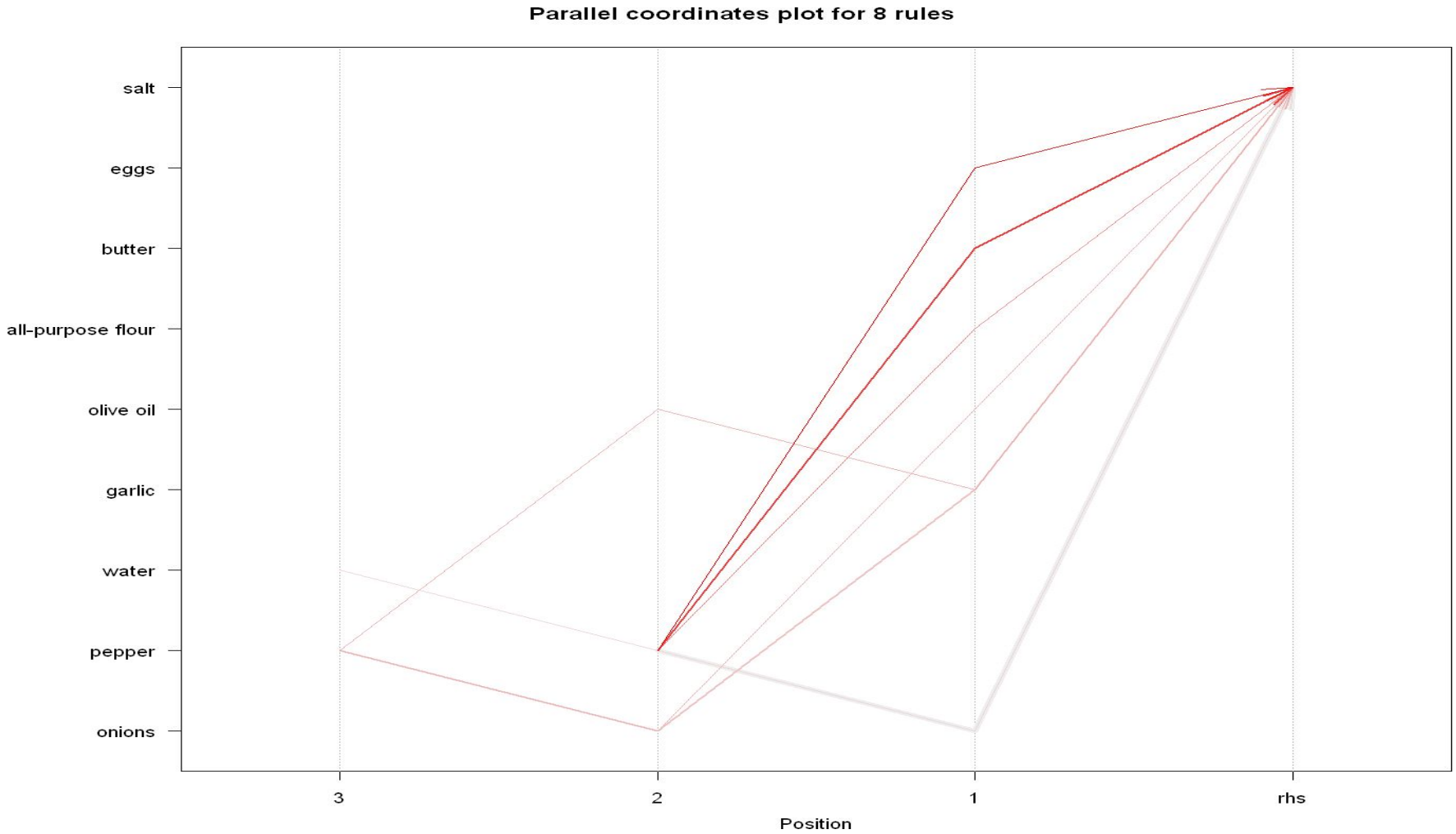

Graph for 8 rules
size: support (0.009 - 0.034)
color: lift (1.988 - 2.053)
all-purpose flour
water
eggs
salt
onions
pepper
olive oil
garlic
butter

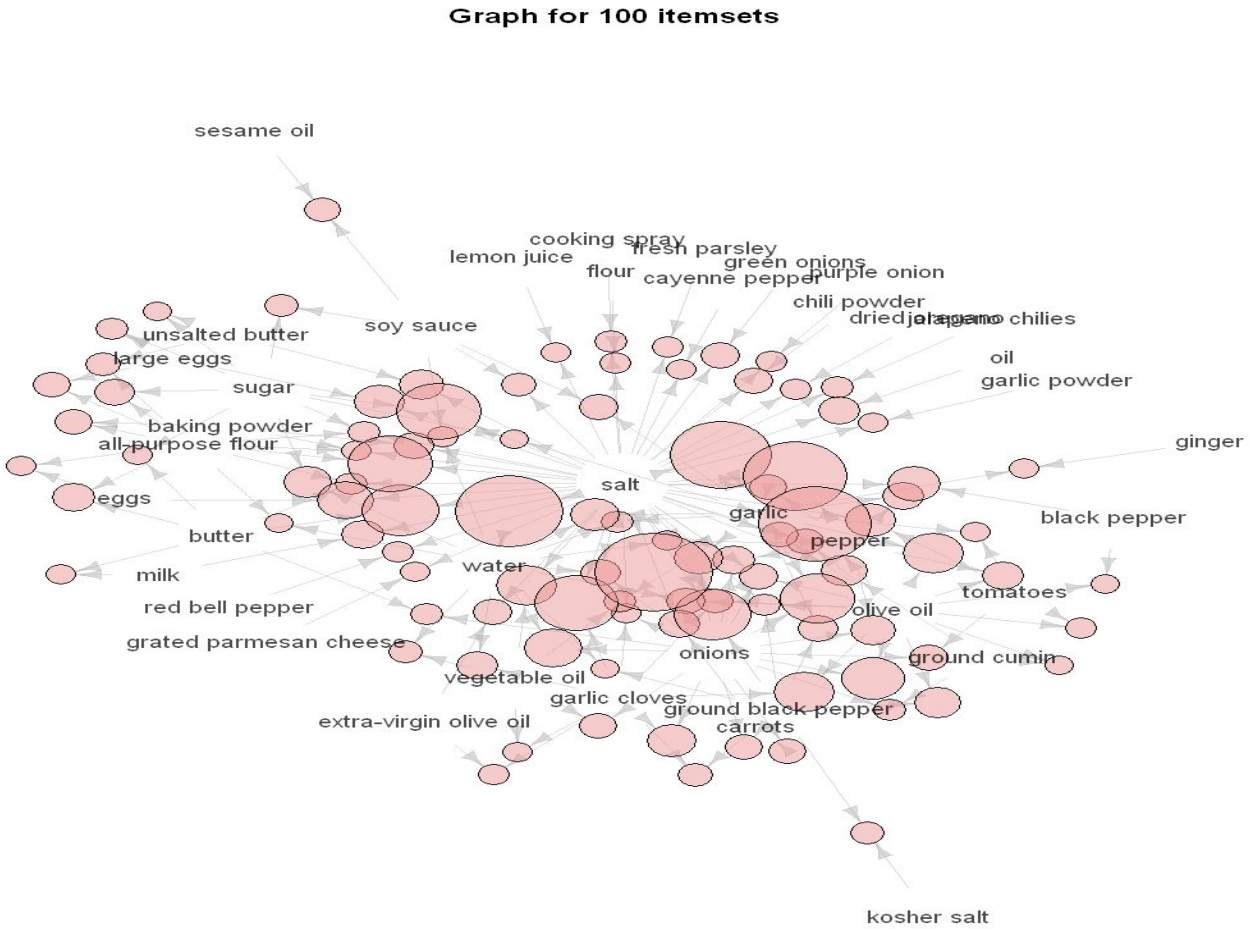

Graph for 8 rules
size: support (0.009 - 0.034)
color: lift (1.988 - 2.053)
onions
olive oil
garlic
pepper
eggs
salt
butter
water
all-purpose flour

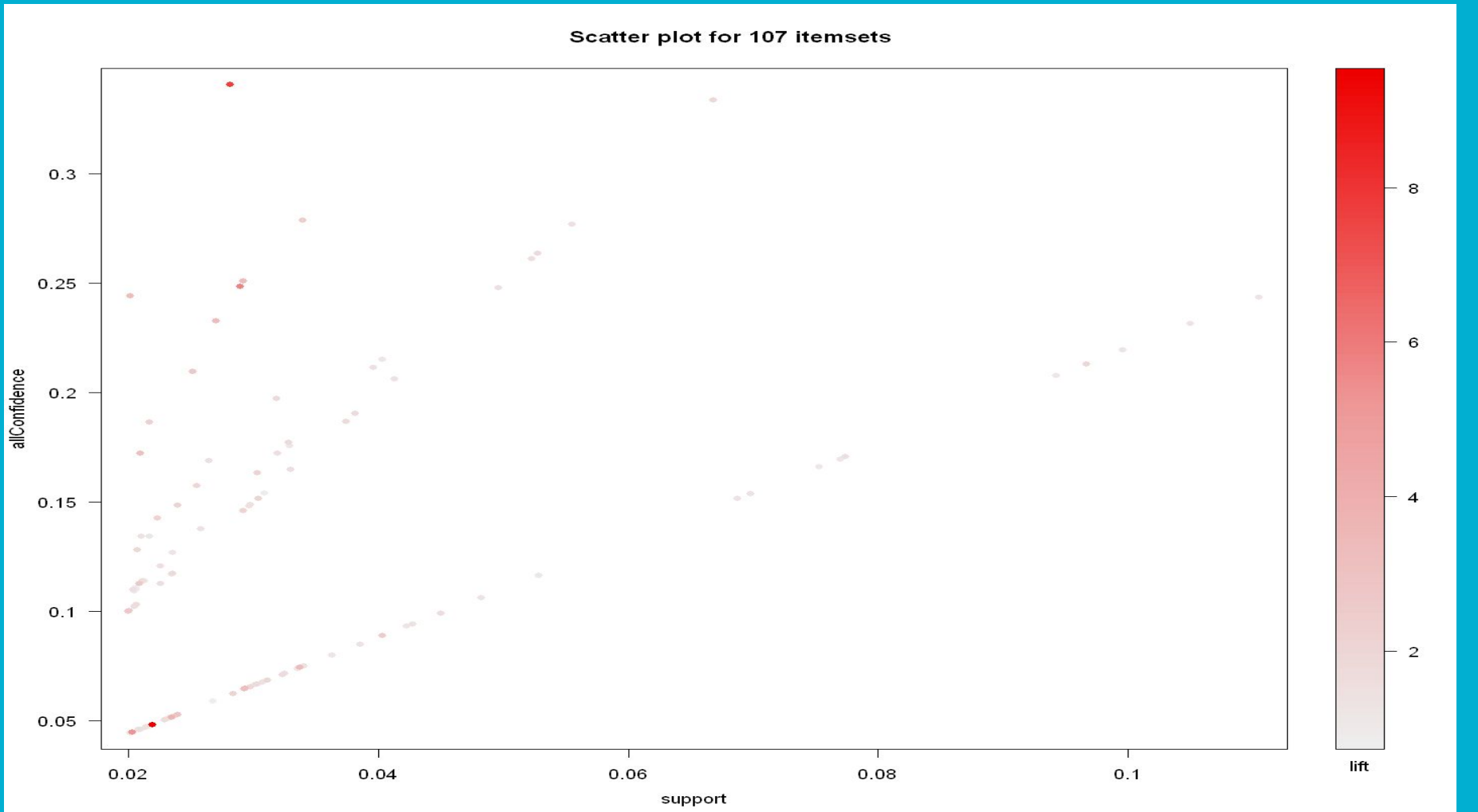

Two-key plot
0.93
0.925
0.92
0.915
0.91
0.905
confidence
0.01
0.015
0.02
0.025
0.03
support
order 4
order 3

Grouped Matrix for 8 Rules

Size: support
Color: lift


Parallel coordinates plot for 8 rules
salt
eggs
butter
all-purpose flour
olive oil
garlic
water
pepper
onions
3
2
1
rhs
Position


Graph for 100 itemsets
size: support (0.02 - 0.11)
sesame oil
cooking spray
fresh parsley
lemon juice
green onions
flour
purple onion
cayenne pepper
chili powder
soy sauce
unsalted butter
large eggs
oil
garlic powder
sugar
baking powder
all-purpose flour
ginger
salt
eggs
garlic
black pepper
butter
pepper
water
milk
tomatoes
red bell pepper
olive oil
grated parmesan cheese
onions
ground cumin
vegetable oil
garlic cloves
ground black pepper
extra-virgin olive oil
carrots
kosher salt


Scatter plot for 107 itemsets
allConfidence
0.3
0.25
0.2
0.15
0.1
0.05
0.02
0.04
0.06
0.08
0.1
support
8
6
4
2
lift

The remaining experimentations and results are shown in the Jupyter Notebook

Clustering Analysis

DATASET

	id	cuisine	ingredients
0	10259	greek	[romaine lettuce, black olives, grape tomatoes...
1	25693	southern_us	[plain flour, ground pepper, salt, tomatoes, g...
2	20130	filipino	[eggs, pepper, salt, mayonaise, cooking oil, g...
3	22213	indian	[water, vegetable oil, wheat, salt]
4	13162	indian	[black pepper, shallots, cornflour, cayenne pe...
...	...	...	...
39769	29109	irish	[light brown sugar, granulated sugar, butter, ...
39770	11462	italian	[KRAFT Zesty Italian Dressing, purple onion, b...
39771	2238	irish	[eggs, citrus fruit, raisins, sourdough starte...
39772	41882	chinese	[boneless chicken skinless thigh, minced garli...
39773	2362	mexican	[green chile, jalapeno chilies, onions, ground...

39774 rows × 3 columns

- **Clustering of Recipes was performed using k-means & Hierarchical methods to separate our recipe dataset into clusters. The number of clusters was chosen using silhouette analysis, choosing k that gave the largest average silhouette score and created clusters with similar sizes (Pedregosa et al., 2011). For speed and accuracy, we used a sparse term frequency inverse document frequency matrix of 30 features. (Ref.:http://cs229.stanford.edu/proj2017/final-reports/5244233.pdf)**

- **These clusters can be used as a means of classifying generated recipes, and we can compare against recipes in the dataset.**

- **To perform clustering, the dataset is prepared in appropriate form as described.**

```
hyphenated_ing_list=[]
for i in col_ingredients:
  li=[]
  for j in i:
    li.append(j.replace(" ","_"))
  hyphenated_ing_list.append(li)

hyphenated_ing_list
```

```
 'carrots',
 'fresh_lime_juice'],
['ground_pepper',
 'half_&_half',
 'fat_skimmed_chicken_broth',
 'fresh_chives',
 'large_eggs',
 'chopped_onion',
 'roasted_red_peppers',
 'salt',
 'orange',
 'potatoes',
 'smoked_trout'],
['fresh_parmesan_cheese',
 'bacon_slices',
 'olive_oil',
 'butternut_squash',
 'salt',
 'ground_black_pepper',
 'purple_onion',
 'swiss_chard',
 'castellane',
```

```
space_ing_list=[]
li=[]
for i in hyphenated_ing_list:
  li.append(' '.join(j for j in i))
  #space_ing_list.append(li)

li
```

```
['romaine_lettuce black_olives grape_tomatoes garlic pepper purple_onion seasoning garbanzo_beans feta_cheese_crumbles',
 'plain_flour ground_pepper salt tomatoes ground_black_pepper thyme eggs green_tomatoes yellow_corn_meal milk vegetable_oil',
 'eggs pepper salt mayonaise cooking_oil green_chilies grilled_chicken_breasts garlic_powder yellow_onion soy_sauce butter chicken_livers',
 'water vegetable_oil wheat salt',
 'black_pepper shallots cornflour cayenne_pepper onions garlic_paste milk butter salt lemon_juice water chili_powder passata oil ground_cumin boneless_chicken_
 'plain_flour sugar butter eggs fresh_ginger_root salt ground_cinnamon milk vanilla_extract ground_ginger powdered_sugar baking_powder',
 'olive_oil salt medium_shrimp pepper garlic chopped_cilantro jalapeno_chilies flat_leaf_parsley skirt_steak white_vinegar sea_salt bay_leaf chorizo_sausage',
 'sugar pistachio_nuts white_almond_bark flour vanilla_extract olive_oil almond_extract eggs baking_powder dried_cranberries',
 'olive_oil purple_onion fresh_pineapple pork poblano_peppers corn_tortillas cheddar_cheese ground_black_pepper salt iceberg_lettuce lime jalapeno_chilies chop
 'chopped_tomatoes fresh_basil garlic extra-virgin_olive_oil kosher_salt flat_leaf_parsley',
 'pimentos sweet_pepper dried_oregano olive_oil garlic sharp_cheddar_cheese pepper swiss_cheese provolone_cheese canola_oil mushrooms black_olives sausages',
 'low_sodium_soy_sauce fresh_ginger dry_mustard green_beans white_pepper sesame_oil scallions canola_oil sugar Shaoxing_wine garlic ground_turkey water crushed_
 'Italian_parsley_leaves walnuts hot_red_pepper_flakes extra-virgin_olive_oil fresh_lemon_juice trout_fillet garlic_cloves chipotle_chile fine_sea_salt flat_le
 'ground_cinnamon fresh_cilantro chili_powder ground_coriander kosher_salt ground_black_pepper garlic plum_tomatoes avocado lime_juice flank_steak salt ground_
 'fresh_parmesan_cheese butter all-purpose_flour fat_free_less_sodium_chicken_broth chopped_fresh_chives gruyere_cheese ground_black_pepper bacon_slices gnocch
 'tumeric vegetable_stock tomatoes garam_masala naan red_lentils red_chili_peppers onions spinach sweet_potatoes',
 'greek_yogurt lemon_curd confectioners_sugar raspberries',
 'italian_seasoning broiler-fryer_chicken mayonaise zesty_italian_dressing',
 'sugar hot_chili asian_fish_sauce lime_juice',
 'soy_sauce vegetable_oil red_bell_pepper chicken_broth yellow_squash garlic_chili_sauce sliced_green_onions broccolini salt fresh_lime_juice cooked_rice chick
 'pork_loin roasted_peanuts chopped_cilantro_fresh hoisin_sauce creamy_peanut_butter chopped_fresh_mint thai_basil rice medium_shrimp water rice_noodles beansp
 'roma_tomatoes kosher_salt purple_onion jalapeno_chilies lime chopped_cilantro',
```

Count Vectorizer

	10_oz	14	14_oz	15_oz	...	zinfandel	ziti	zucchini	zucchini_blossoms
0	0	0	0	0	...	0	0	0	0
1	0	0	0	0	...	0	0	0	0
2	0	0	0	0	...	0	0	0	0
3	0	0	0	0	...	0	0	0	0
4	0	0	0	0	...	0	0	0	0
...	...	..	...	...	...	...	...	...	...
39769	0	0	0	0	...	0	0	0	0
39770	0	0	0	0	...	0	0	0	0
39771	0	0	0	0	...	0	0	0	0
39772	0	0	0	0	...	0	0	0	0
39773	0	0	0	0	...	0	0	0	0

[39774 rows x 6854 columns]

TD-IDF Vectorizer

	10_oz	14	14_oz	15_oz	...	zinfandel	ziti	zucchini	zucchini_blossoms
0	0.0	0.0	0.0	0.0	...	0.0	0.0	0.0	0.0
1	0.0	0.0	0.0	0.0	...	0.0	0.0	0.0	0.0
2	0.0	0.0	0.0	0.0	...	0.0	0.0	0.0	0.0
3	0.0	0.0	0.0	0.0	...	0.0	0.0	0.0	0.0
4	0.0	0.0	0.0	0.0	...	0.0	0.0	0.0	0.0
...	...	...	...	...	...	...	...	...	...
39769	0.0	0.0	0.0	0.0	...	0.0	0.0	0.0	0.0
39770	0.0	0.0	0.0	0.0	...	0.0	0.0	0.0	0.0
39771	0.0	0.0	0.0	0.0	...	0.0	0.0	0.0	0.0
39772	0.0	0.0	0.0	0.0	...	0.0	0.0	0.0	0.0
39773	0.0	0.0	0.0	0.0	...	0.0	0.0	0.0	0.0

We cannot apply PCA on CSR Matrix. In such a case, we need to use TruncatedSVD function for dimension reduction.

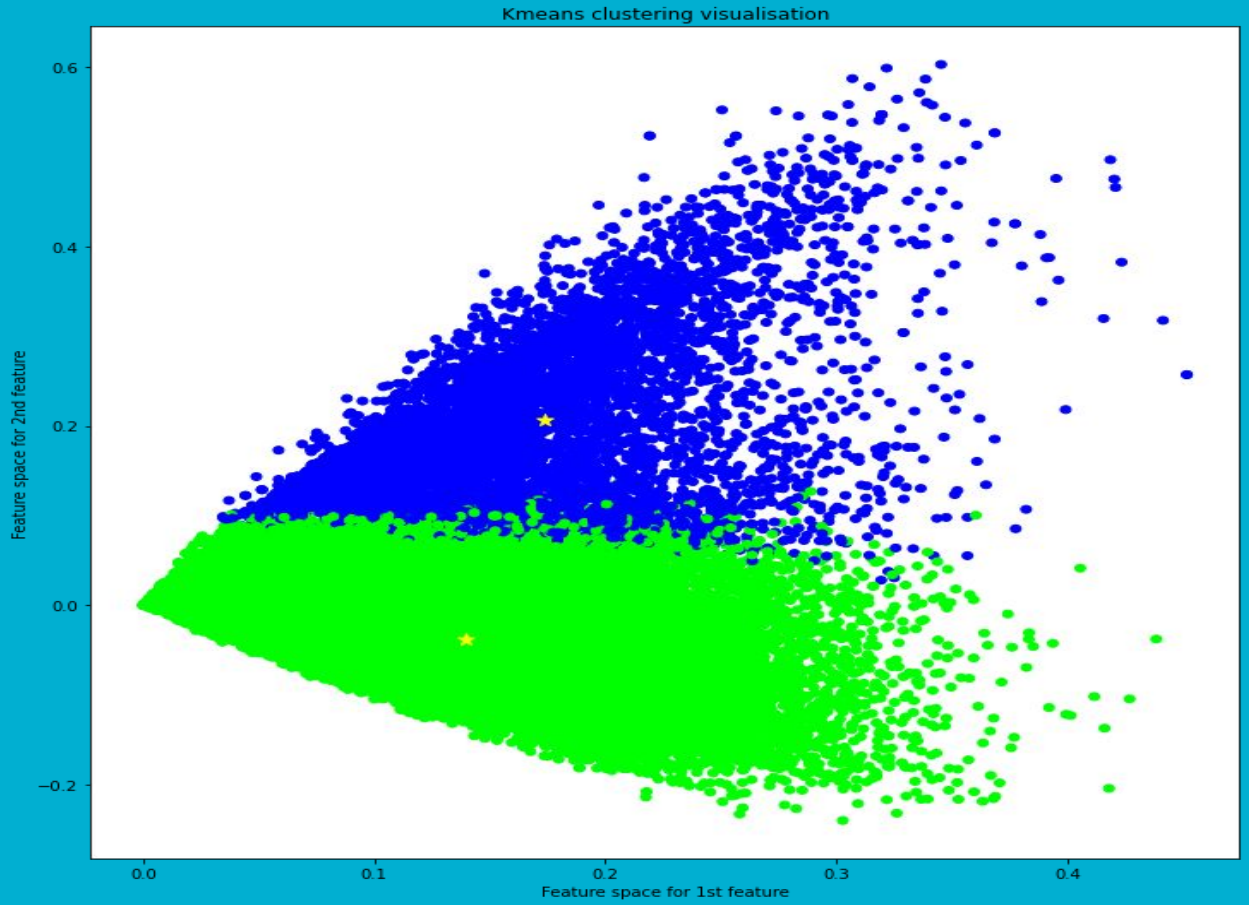
Then silhoutte coefficient analysis was performed to find the optimal value of k.

```
# Finding the optimum k value for kmeans
sil=[]
kmax=10

for k in range(2,kmax+1):
  kmeans = KMeans(n_clusters =k, random_state = 0).fit(X_svd)
  labels = kmeans.labels_
  sil.append(silhouette_score(X_svd,labels))

sil
```

```
[0.15421126533831006,
 0.14010055967277107,
 0.0920665064818288,
 0.08930986252138934,
 0.10004595635792736,
 0.08698403105788396,
 0.08391461512162042,
 0.08235271018400023,
 0.08790896990420013]
```

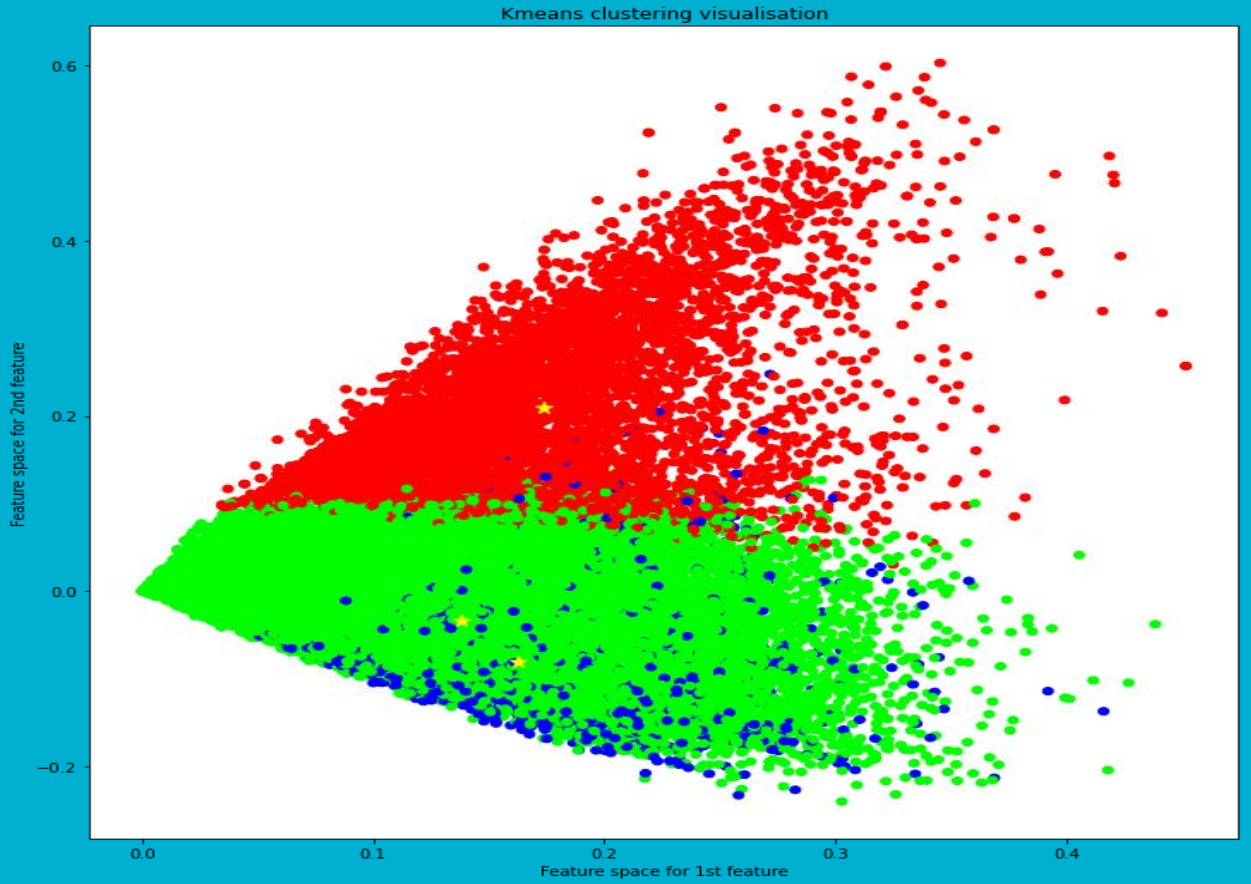
KMEANS CLUSTERING

With k=2



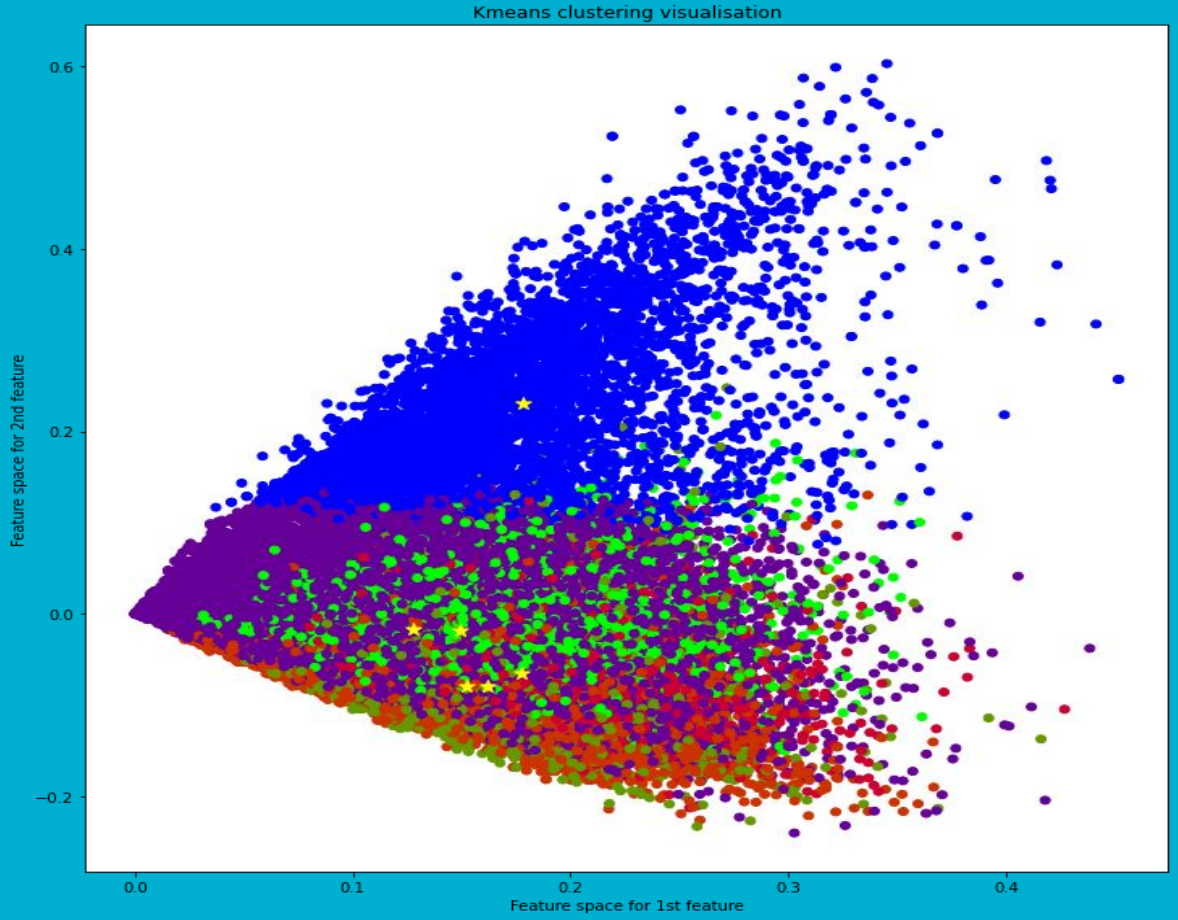
KMEANS CLUSTERING

With k=3



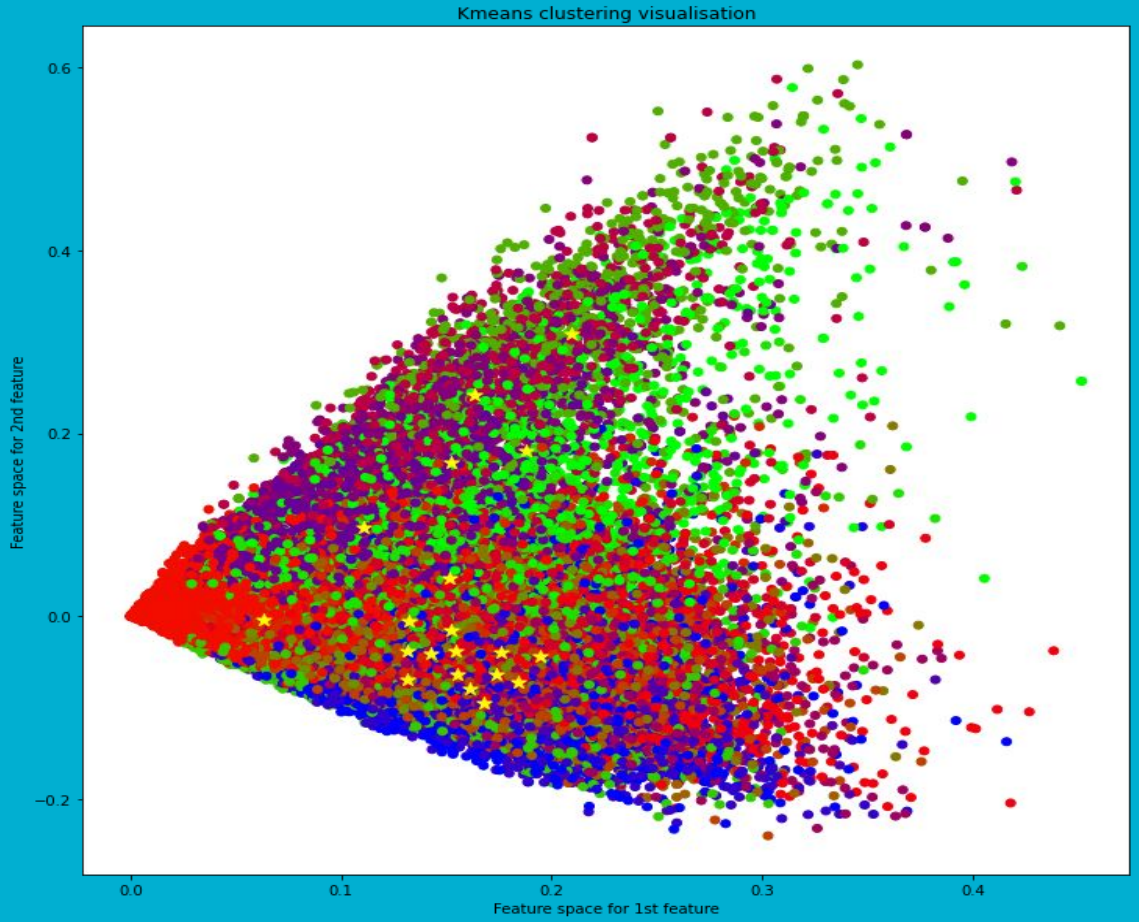
KMEANS CLUSTERING

With k=6



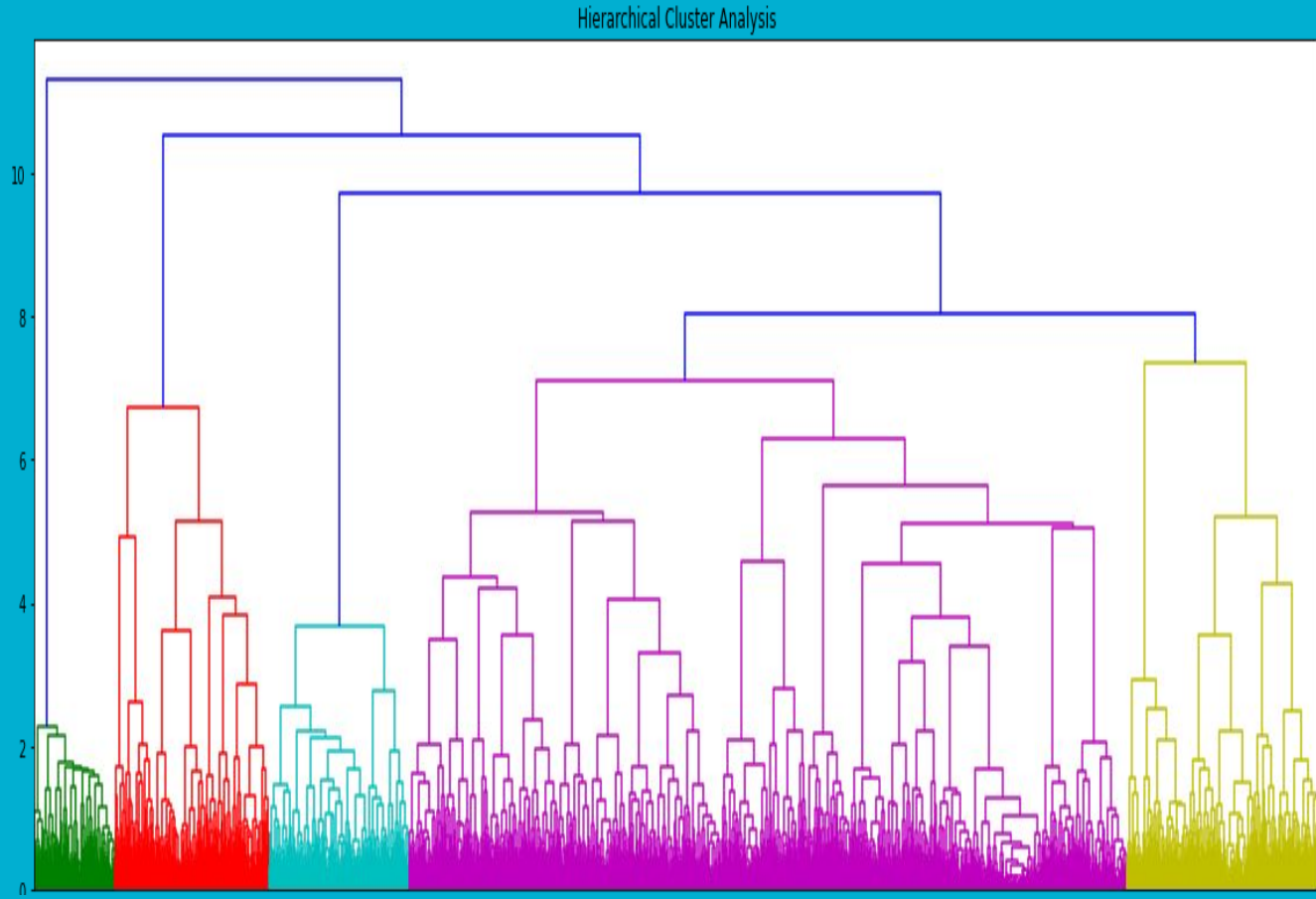
KMEANS CLUSTERING

With k=20



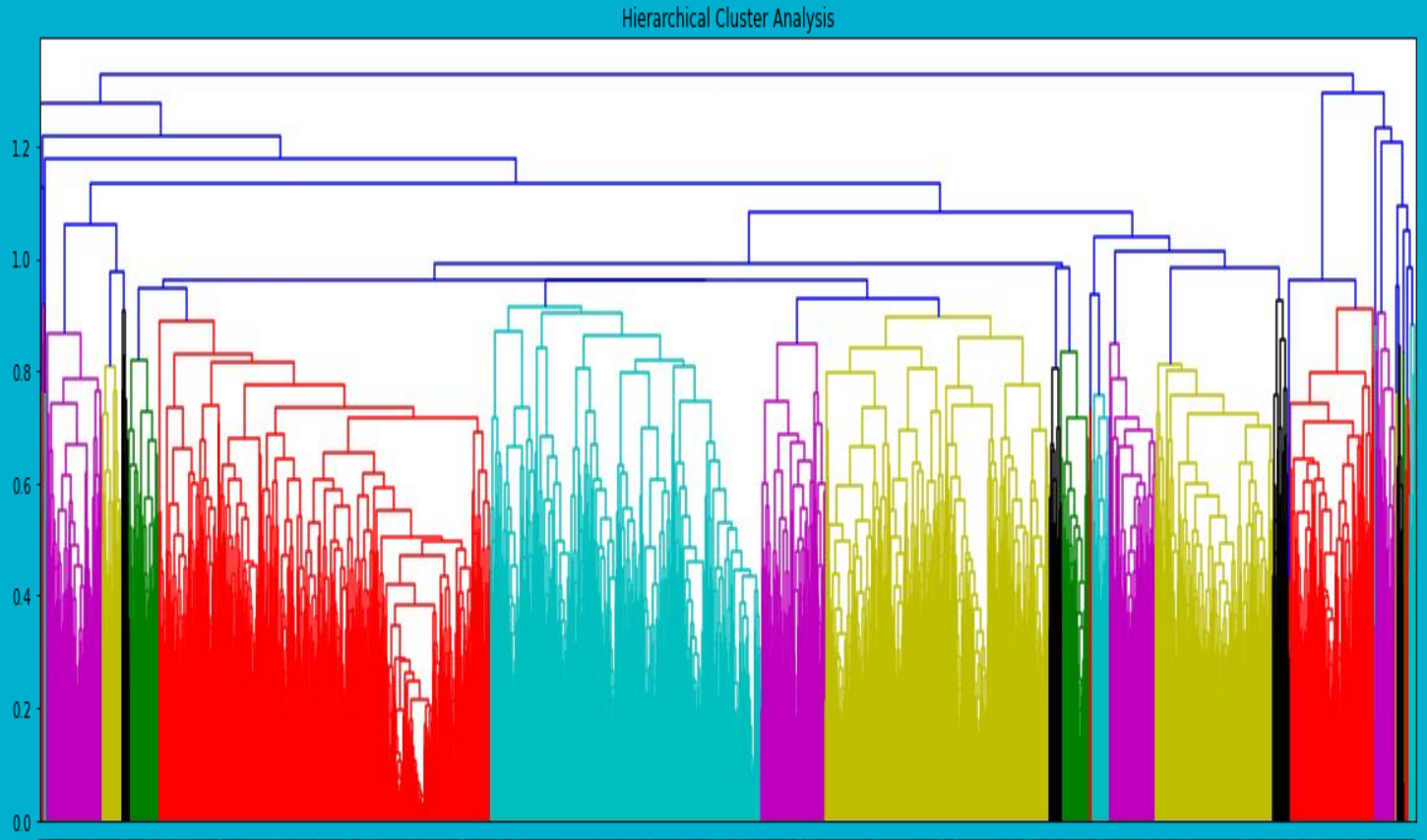
HIERARCHICAL CLUSTERING

With “ WARD METHOD LINKAGE”



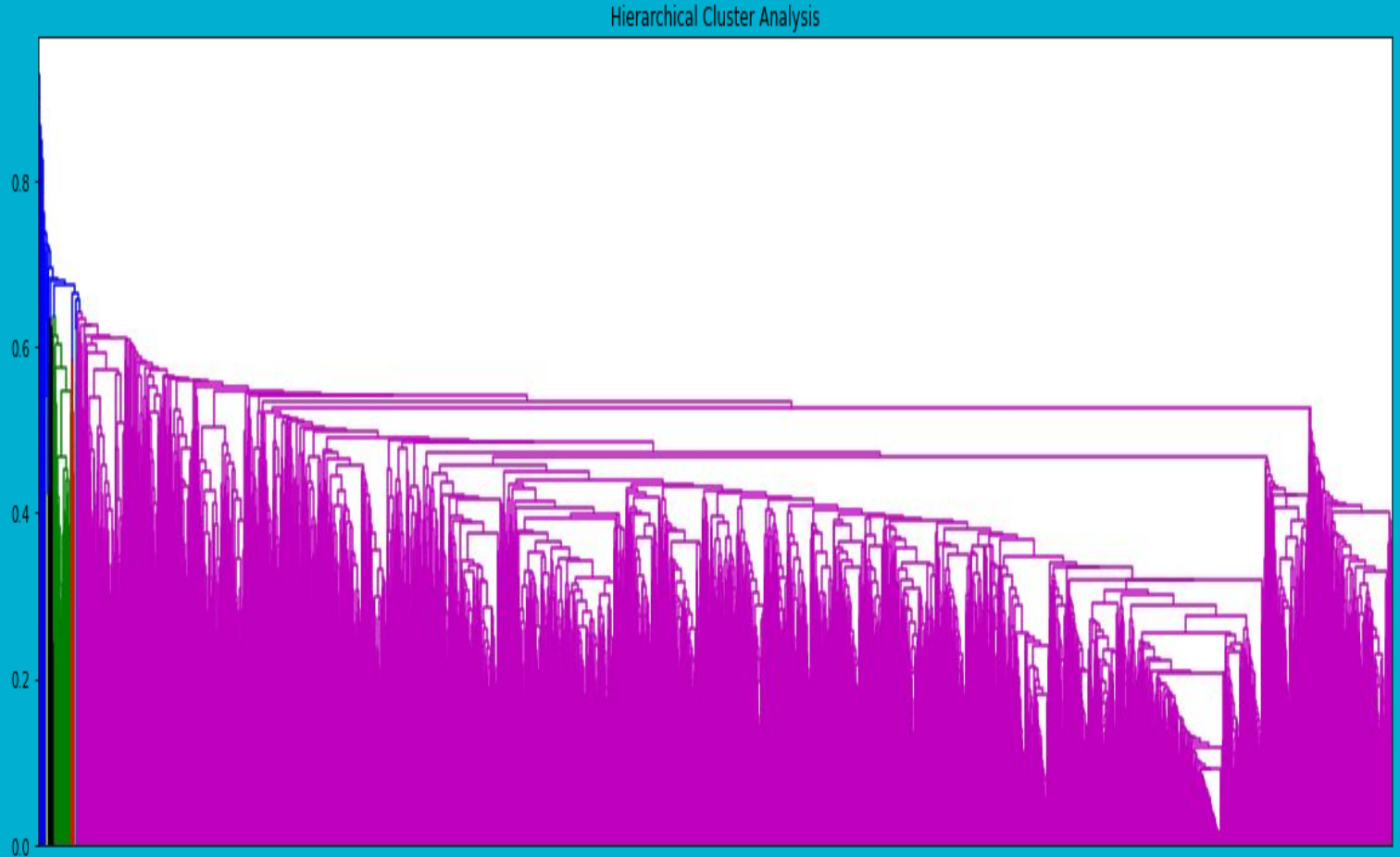
HIERARCHICAL CLUSTERING

With “ COMPLETE LINKAGE”



HIERARCHICAL CLUSTERING

With “ GROUP AVERAGE LINKAGE”



THANK YOU