TASK REPORT
DATA SCIENCE AND ITS IMPLEMENTATION IN MACHINE LEARNING

By:
Mohamad Irwan Afandi
Data Science Student

Data Fellowship
2020

Chapter 1
Introduction

A *customer is* a person or company that receives, consumes or buys a product or service and can choose between different goods and suppliers. Generally the customer will back to us we they feel satisfactory service. This is why we need to know our customer behavior to determine what should we do to our customer in the future. To know the customer behavior, we can use customer segmentation and market basket analysis. In this report I want to show you how to get the customer behavior based on the problem below.

Suppose that you are working in a consulting company. You meet a client who is interested in implementing a data science project for the first time to solve their problem. You are provided with some problems to be solved, just to examine your capabilities.
Customer Segmentation and Market Basket Analysis:
https://www.kaggle.com/mgmarques/customer-segmentation-and-market-basket-analysis/data .

Do not forget to implement CRISP-DM steps! Please be creative and hopefully, you can implement the best coding practices in your code.

Chapter 2
Progress Report

In this chapter you will have to fill in the table below according to the progress of the project that you have made along the way. We need to know how long it takes for you and how big the effort that you have done in order to complete this task. We appreciate detailed information.

Day/Date	**Task**	**Level** (easy/medium/hard)	**Comments**
25/08/2020	Doing ML quizzes in iykra's website and try to preprocessing the practice case data	Medium	There are a lot of subject that need to understand.
26/08/2020	Doing ML quizzes in iykra's website, build a cluster and apriori.	Hard	I never build Kmeans and Apriori with Association Rule before. I try to contact the mentor, but does not get any response. So I don't know the customer segmentation is true of false.
27-30/08/2020	ML report, PPT and Video.	Medium	Wew, spend a lot of times.

Chapter 3
Task Report

There are a lot of problems in retail that can be solve by machine learning like customer segmentation, market basket analysis, prize optimization & promotions, optimized route planning, etc. In this report, I will explain how to build model of customer segmentation and market basket analysis on machine learning. To make this process easier to understand, I use CRISP-DM framework. This process start from business understanding, data understanding, data preparation, modeling, evaluation, and deployment (doesn't implement this step). Below is the result from this modeling process.

Business Understanding

Before we try to build the model, we need to know what is customer segmentation and market basket analysis.

- Customer segmentation: Customer segmentation is the problem of uncovering information about a firm's customer base, based on their interactions with the business. In most cases this interaction is in terms of their purchase behavior and patterns. Based on this problem, we can categorize the customers on several groups. For example we grouped them into 4 groups like defector, mercenary, hostage and loyalist. After we know every customer categorization, we can define what should we do to make the mercenary's customer to be loyalist, etc.
- Market basket analysis: Market basket analysis is a method to gain insights into granular behavior of customers. This is helpful in devising strategies which uncovers deeper understanding of purchase decisions taken by the customers. This is interesting as a lot of times even the customer will be unaware of such biases or trends in their purchasing behavior. After we know the customer behavior, we can rearrange the product place to increase the product sales.

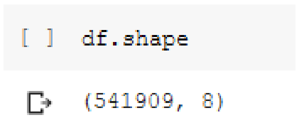
Data Understanding

In this case I used retail data with excel file format that I have downloaded from Kaggle (https://www.kaggle.com/mgmarques/customer-segmentation-and-market-basket-analysis/data). Below is the dictionary of the data.

• InvoiceNo	: A unique identifier for the invoice. An invoice number shared across rows means that those transactions were performed in a single invoice (multiple purchases).
• StockCode	: Identifier for items contained in an invoice.
• Description	: Textual description of each of the stock item.

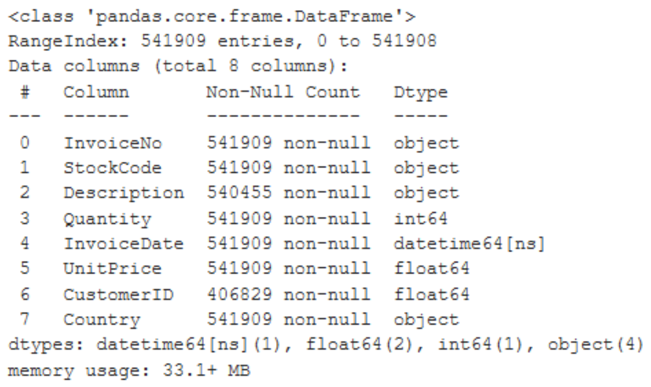
- Quantity : The quantity of the item purchased.
- InvoiceDate : Date of purchase.
- UnitPrice : Value of each item.
- CustomerID : Identifier for customer making the purchase.
- Country : Country of customer.

With pandas, we can also see the dimension of the data and also its data type. Below is the detail of our retail data.



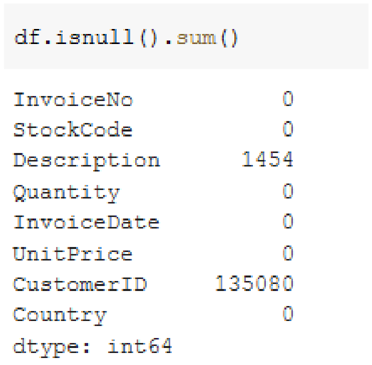
```
[ ]  df.shape

(541909, 8)
```

Using df.shape, we can see that the dimension of our data is 541909 x 8 where the data has 541909 rows and 8 features.



```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 541909 entries, 0 to 541908
Data columns (total 8 columns):
 #   Column       Non-Null Count   Dtype
---  ------       --------------   -----
 0   InvoiceNo    541909 non-null  object
 1   StockCode    541909 non-null  object
 2   Description  540455 non-null  object
 3   Quantity     541909 non-null  int64
 4   InvoiceDate  541909 non-null  datetime64[ns]
 5   UnitPrice    541909 non-null  float64
 6   CustomerID   406829 non-null  float64
 7   Country      541909 non-null  object
dtypes: datetime64[ns](1), float64(2), int64(1), object(4)
memory usage: 33.1+ MB
```

Using df.info, we can see all the data features with its data type and its number of non-null value. There are 3 data types that is object (InvoiceNo, StocekCode, Description and Country), numeric (Quantity, UnitePrice, and CustomerID), and datetimes64 (InvoiceDate). Based on that data we also know that there are 2 feature with missing values that is Description and CustomerID. So we need to handle in the next section.

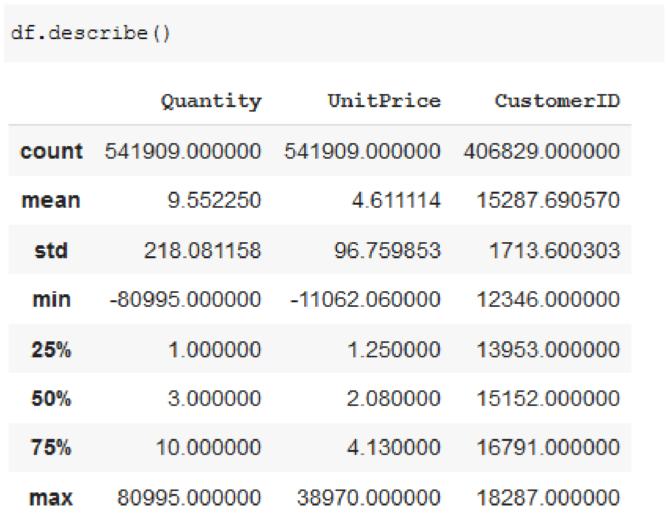


```
df.isnull().sum()

InvoiceNo          0
StockCode          0
Description     1454
Quantity           0
InvoiceDate        0
UnitPrice          0
CustomerID    135080
Country            0
dtype: int64
```

Data Preparation

In this dataset we have 541.909 data with some problems like missing value, data anomalies and count of data. So what must we do is handle all problems in the data, and below what I have done whit this data.

- Look at the data statistical summary

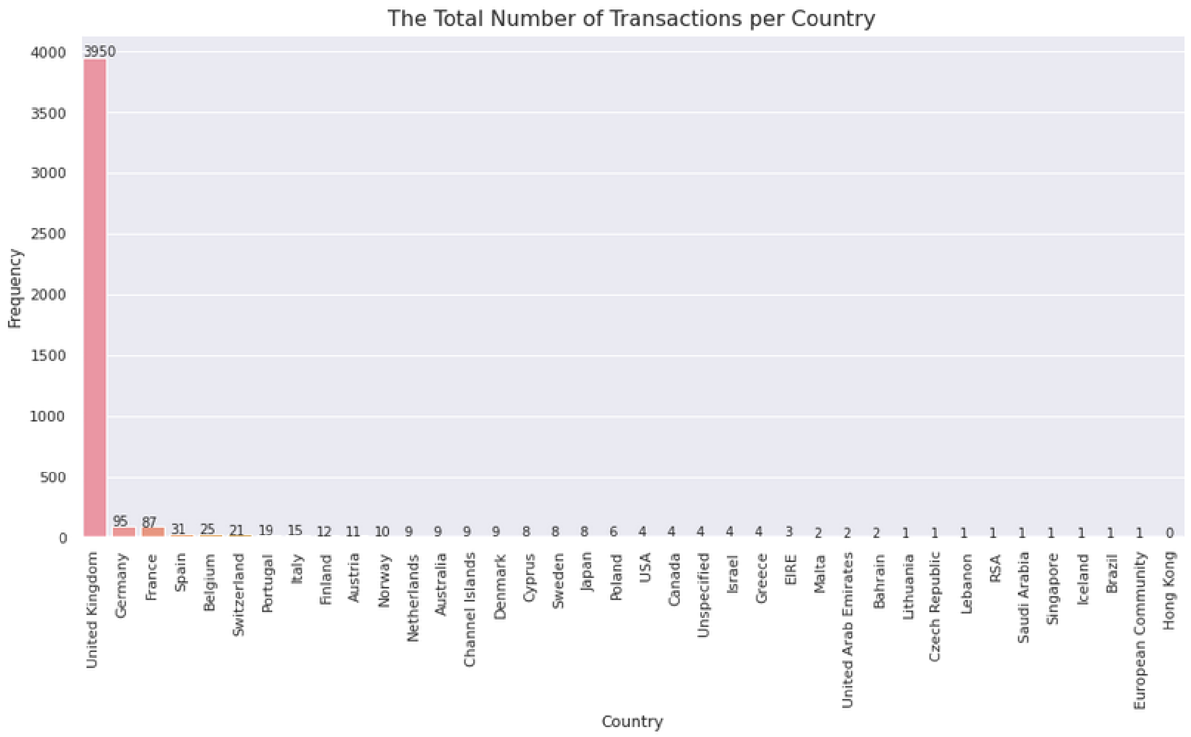


```
df.describe()
```

	Quantity	UnitPrice	CustomerID
count	541909.000000	541909.000000	406829.000000
mean	9.552250	4.611114	15287.690570
std	218.081158	96.759853	1713.600303
min	-80995.000000	-11062.060000	12346.000000
25%	1.000000	1.250000	13953.000000
50%	3.000000	2.080000	15152.000000
75%	10.000000	4.130000	16791.000000
max	80995.000000	38970.000000	18287.000000

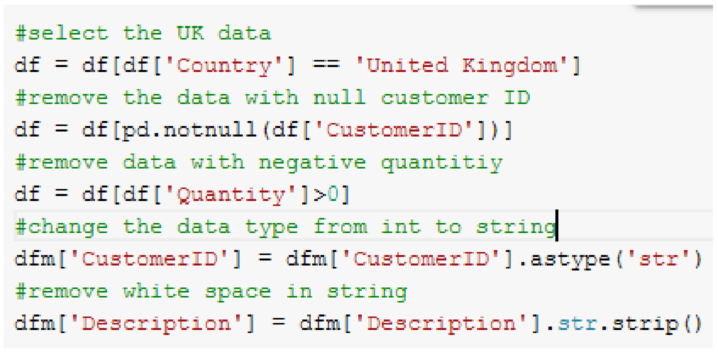
Based on that data, there are two features that has abnormal data: Quantity and UnitPrice because both of the min values are minus. So we need to filter the data and drop the data with minus min value. Another thing we have to do is change the data type of CustomerID into string, because actually it is categorical variable. But before we drop the data, I only use the data from the country with the highest number of transactioins.

- Find the number of unique transaction in every country, and choose the most one. A Larger amounts of data will make our model better. Based this filter I choose United Kingdom with 3950 transaction as my data.



- Filter the data based on this condition:
  - Country is United Kingdom.
  - Remove data without CustomerID.
  - Remove the data with minus quantity value (use describe to see the min value).
  - Remove the data that contain “C” character in its InvoceNo.
  - Remove white space before and after description.

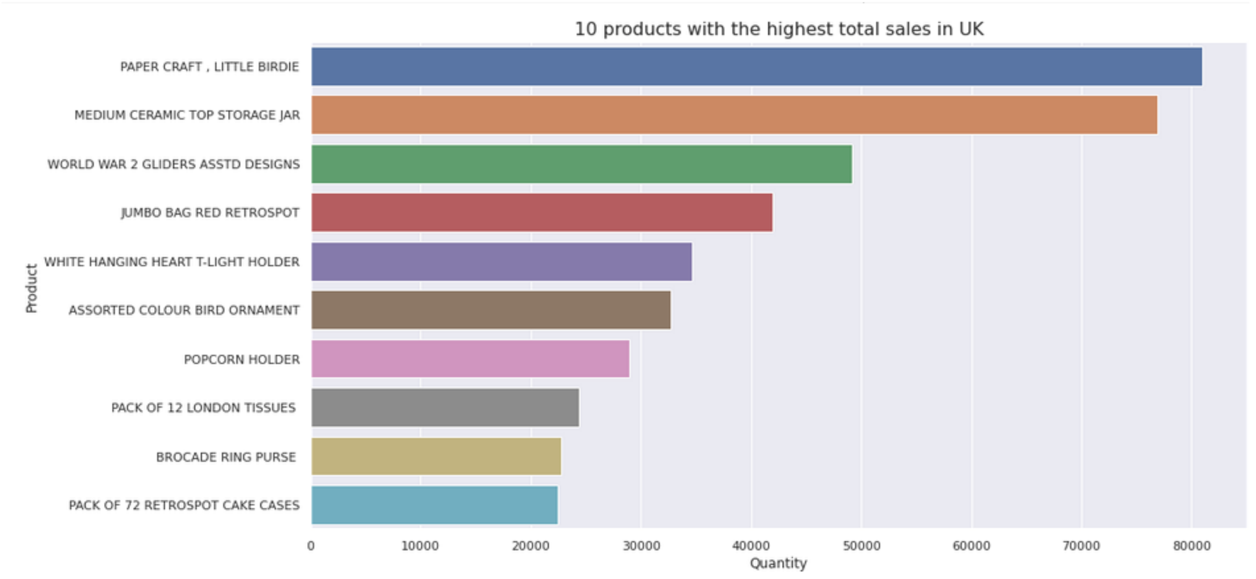
  After doing this step, the remaining of transaction amount is 354.354. Your data is ready to use now.



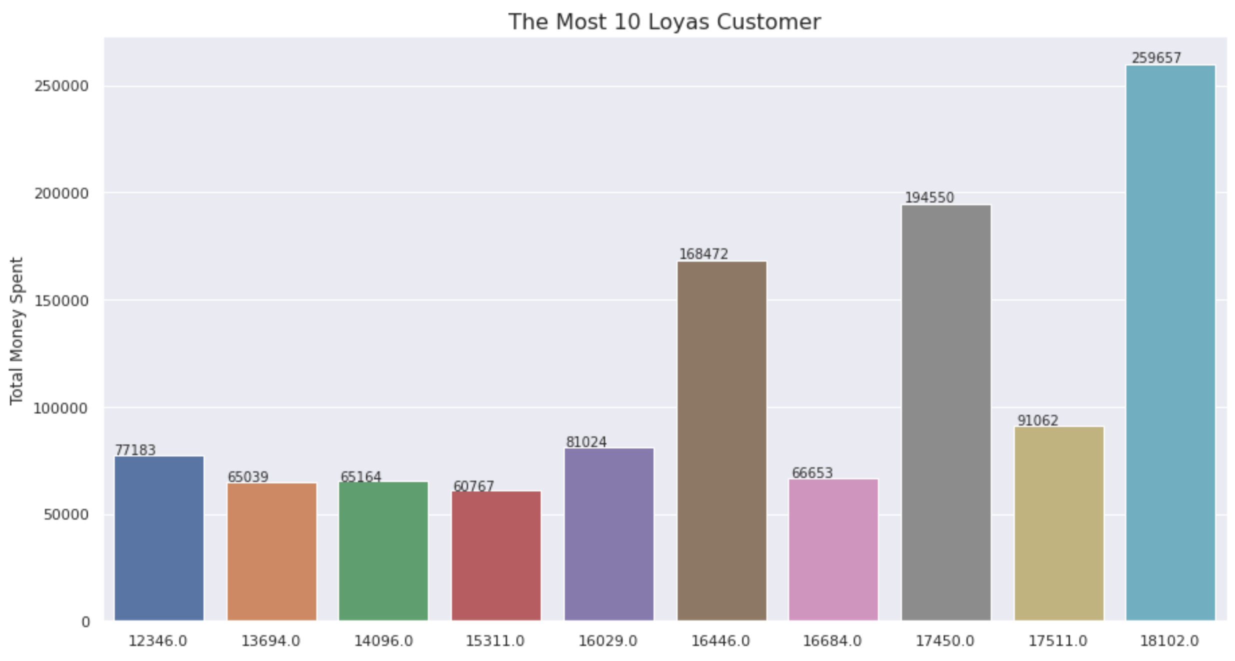
```
#select the UK data
df = df[df['Country'] == 'United Kingdom']
#remove the data with null customer ID
df = df[pd.notnull(df['CustomerID'])]
#remove data with negative quantitiy
df = df[df['Quantity']>0]
#change the data type from int to string
dfm['CustomerID'] = dfm['CustomerID'].astype('str')
#remove white space in string
dfm['Description'] = dfm['Description'].str.strip()
```

Try to do EDA on the data

1. Get the list of best seller product



2. Get the list of loyal customer



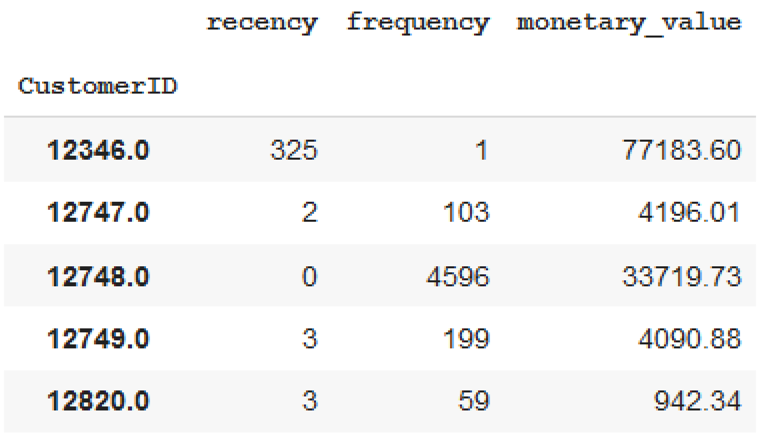
Modeling & Evaluation

Customer Segmentation

To get more insights from the transaction data, I use RFM (Recency, Frequency, and Monetary) method and then use K-Means clustering to categorize the customer. This is the step to get each customers categorization.

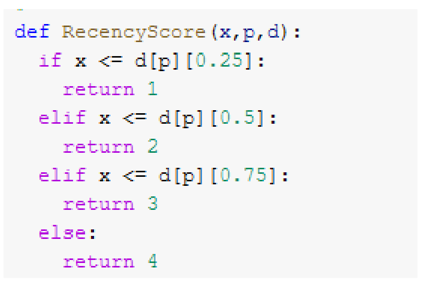
- Group the data based on 'CustomerID' (we need categorize every customers, not every invoice), then calculate the value RFM's value. To calculate the RFM value, you can use this formula.
  - Recency = time.now(max(InvoiceDate)+1) - the last date the customer purchased our products. To know whether he is sleep customer or not.
  - Frequency = count(InvoiceNo) group by CustomerID, we want to know how often a customer make transactions for our products.
  - Monetary = sum(TotalPrice) group by customerID, we want to know how much they spend their money for our products.

  Below is the result from this process

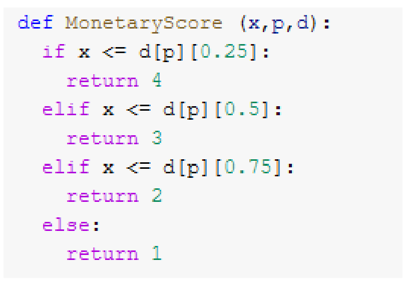


CustomerID	recency	frequency	monetary_value
12346.0	325	1	77183.60
12747.0	2	103	4196.01
12748.0	0	4596	33719.73
12749.0	3	199	4090.88
12820.0	3	59	942.34

- After we get that result, we need to scale it in some groups but in this case you cannot use standardization or normalization. You can use quantile method by dividing that data into 4 group. Set the value of 1 if that is the best value, and 4 if that is worst value. You can implement using this syntax.



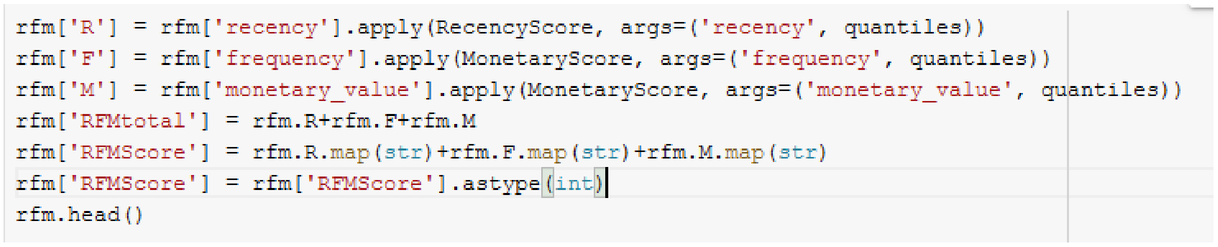
```
def RecencyScore(x,p,d):
  if x <= d[p][0.25]:
    return 1
  elif x <= d[p][0.5]:
    return 2
  elif x <= d[p][0.75]:
    return 3
  else:
    return 4
```



```
def MonetaryScore (x,p,d):
  if x <= d[p][0.25]:
    return 4
  elif x <= d[p][0.5]:
    return 3
  elif x <= d[p][0.75]:
    return 2
  else:
    return 1
```

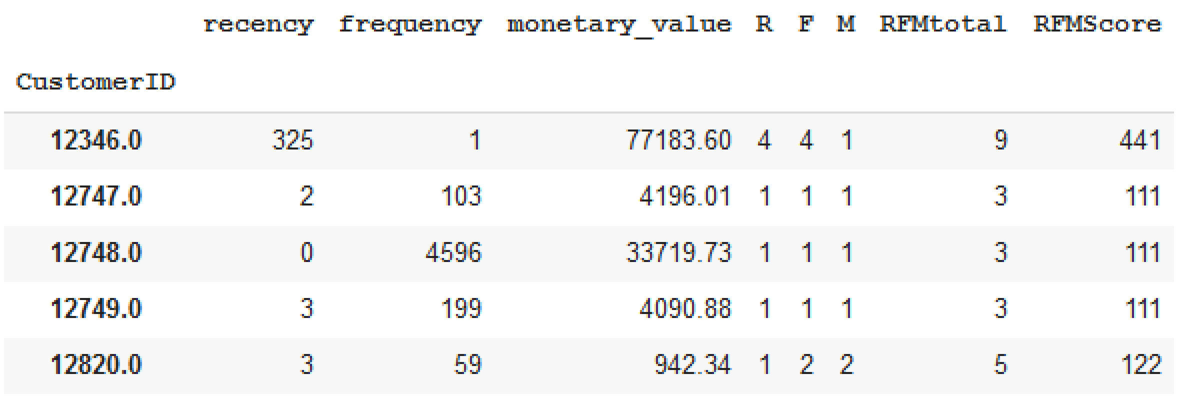
Note: if the recency value is little, it means they just bought our products (set 1) but if the recency value is a lot, it means they are sleep customer (set 4). It's different with frequency and monetary, if the value is a lot it is better. Below is the result of RFM for this data.

To binning the data we just need to call the function above for our recency, frequency and monetary value. Use the syntax below to get the RMF value.



```
rfm['R'] = rfm['recency'].apply(RecencyScore, args=('recency', quantiles))
rfm['F'] = rfm['frequency'].apply(MonetaryScore, args=('frequency', quantiles))
rfm['M'] = rfm['monetary_value'].apply(MonetaryScore, args=('monetary_value', quantiles))
rfm['RFMtotal'] = rfm.R+rfm.F+rfm.M
rfm['RFMScore'] = rfm.R.map(str)+rfm.F.map(str)+rfm.M.map(str)
rfm['RFMScore'] = rfm['RFMScore'].astype(int)
rfm.head()
```

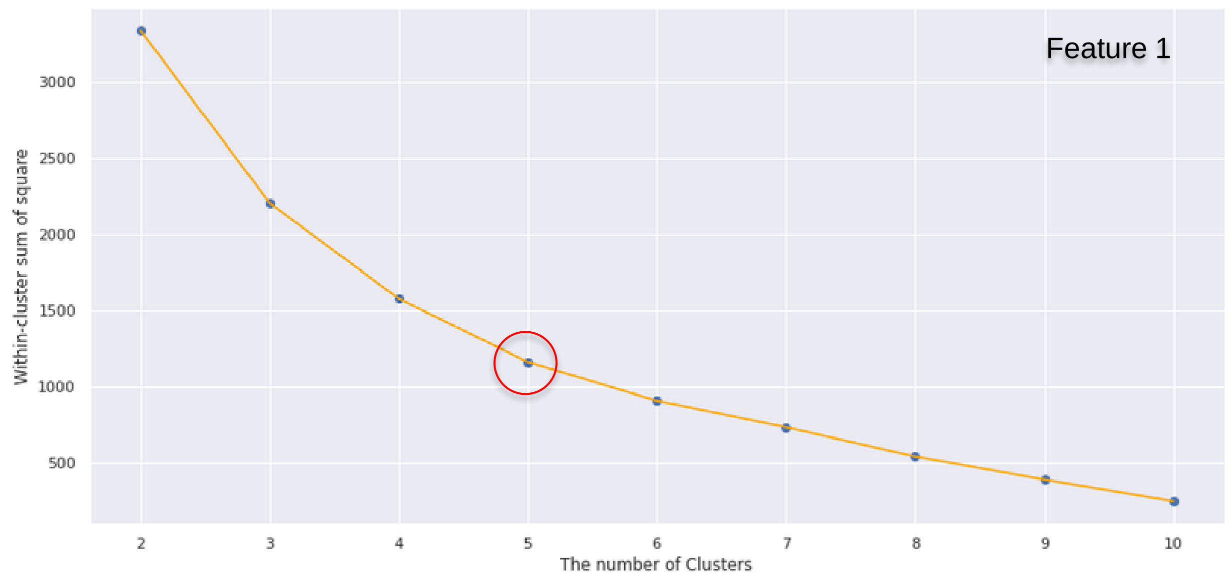
The picture below is the result of the syntax above is.

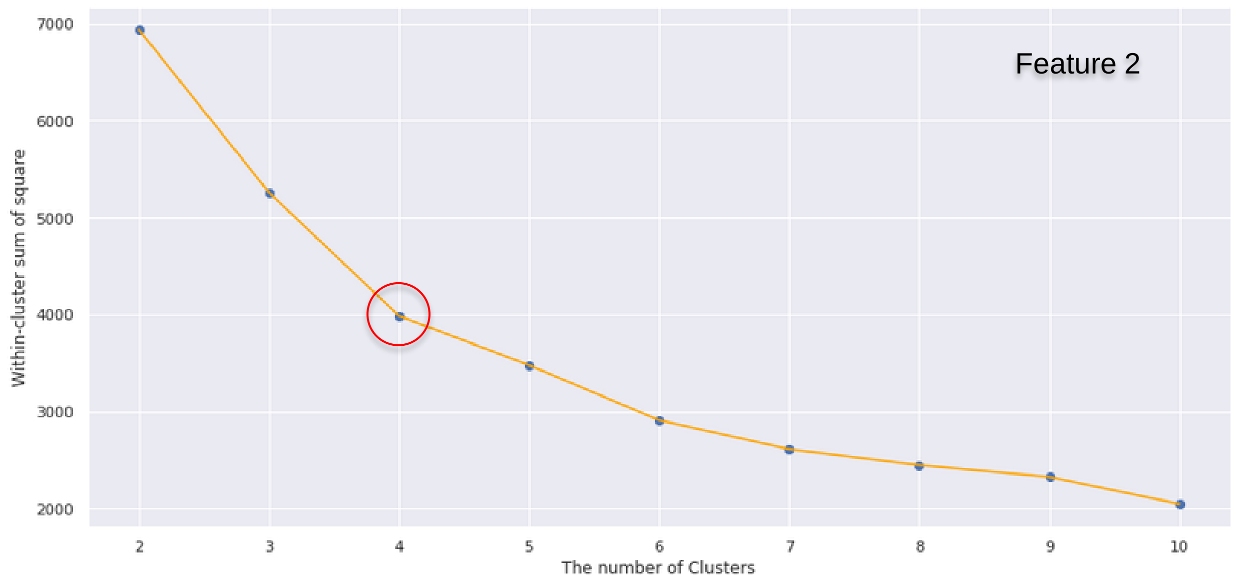


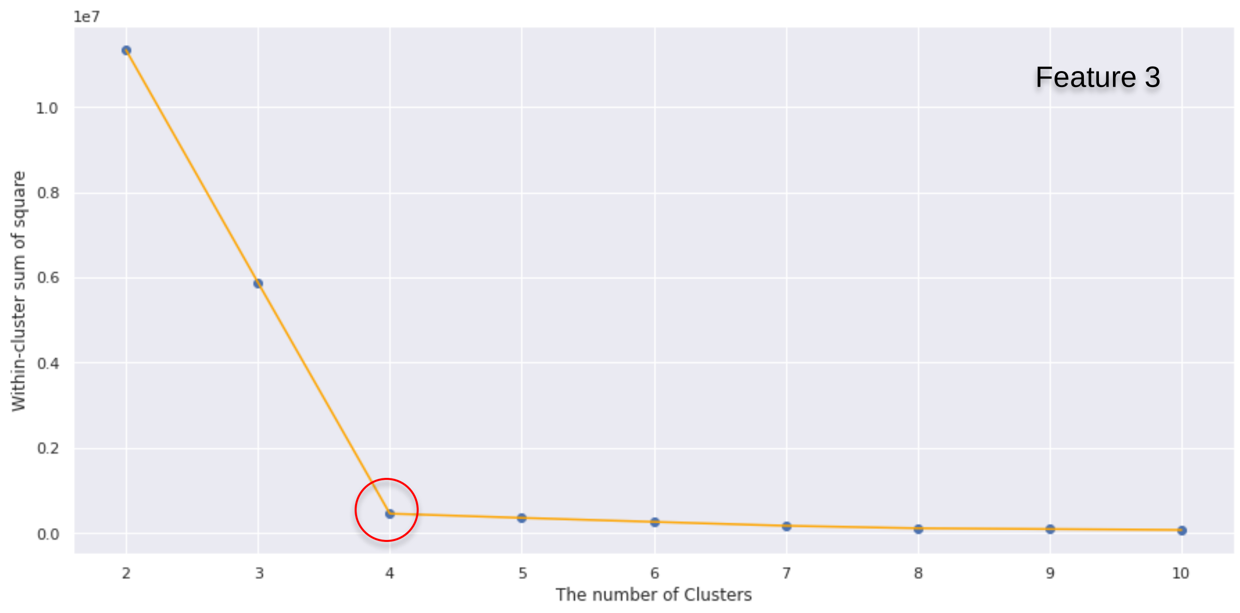
	recency	frequency	monetary_value	R	F	M	RFMtotal	RFMScore
CustomerID								
12346.0	325	1	77183.60	4	4	1	9	441
12747.0	2	103	4196.01	1	1	1	3	111
12748.0	0	4596	33719.73	1	1	1	3	111
12749.0	3	199	4090.88	1	1	1	3	111
12820.0	3	59	942.34	1	2	2	5	122

- Choose the feature to train the data, in this case I use 3 scenarios to select the features.
  - Feature1 : F,M (If you just want to know the customer based on his loyalty just use this feature)
  - Feature2 : R, F, and M (If you want to know the customer based on his loyalty and satisfaction, you can add with recency. Why we need this feature, because if someone feel cozy with the service they will still (often) buy items at the same place, so the recency is little).
  - Feature3 : R, F, M, and RFMScore.
- Train every features with K-Means and choose the best K parameter with WCSS (Within-cluster sum of square) to get the optimum cluster model. To determine which

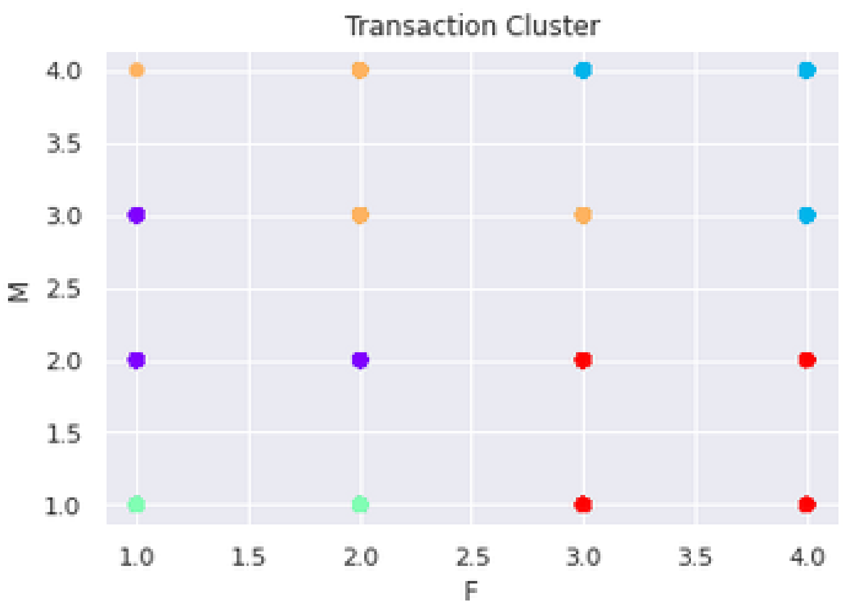
K is the best, choose the last K that wcss value is not too different from the next wcss K value. (Evaluation Process)



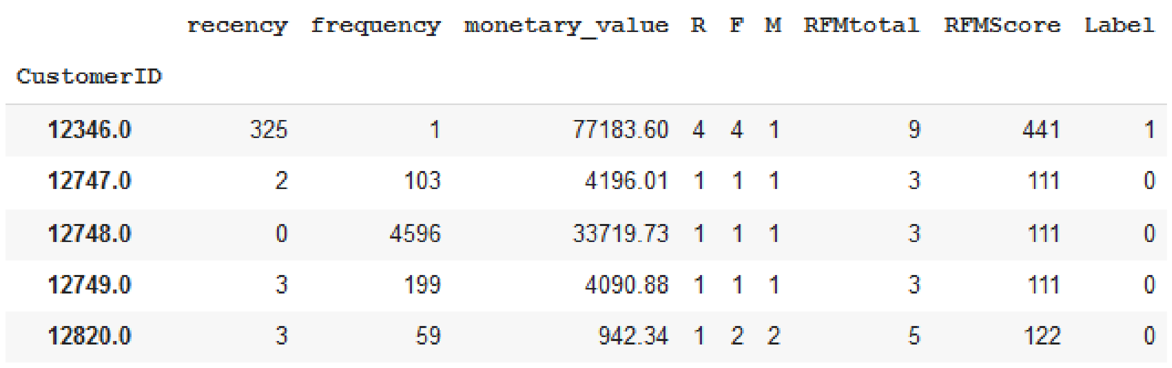




- Train the model K= 4 or 5 but I rather 4 than 5 because both of feature2 and feature3 use K=4 (Choose the number of K is the kind of evaluating process). Then get the label from the cluster model. You can also plot the data if the data use 2 feature to make the visualization of the data. If it's more than 2 features, just save the cluster in the data frame. Below is the visualization example.

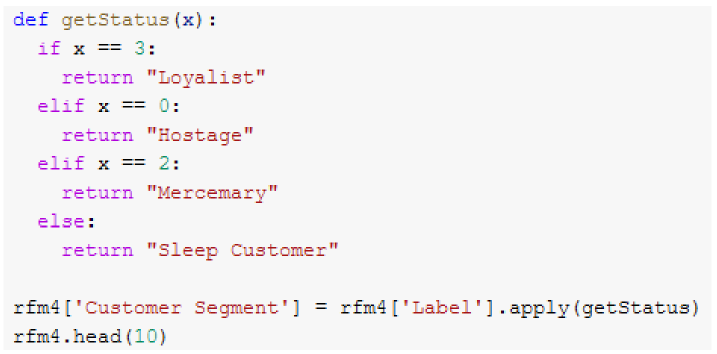


Below is the data frame result, where label is the cluster of the customer (Feature3)



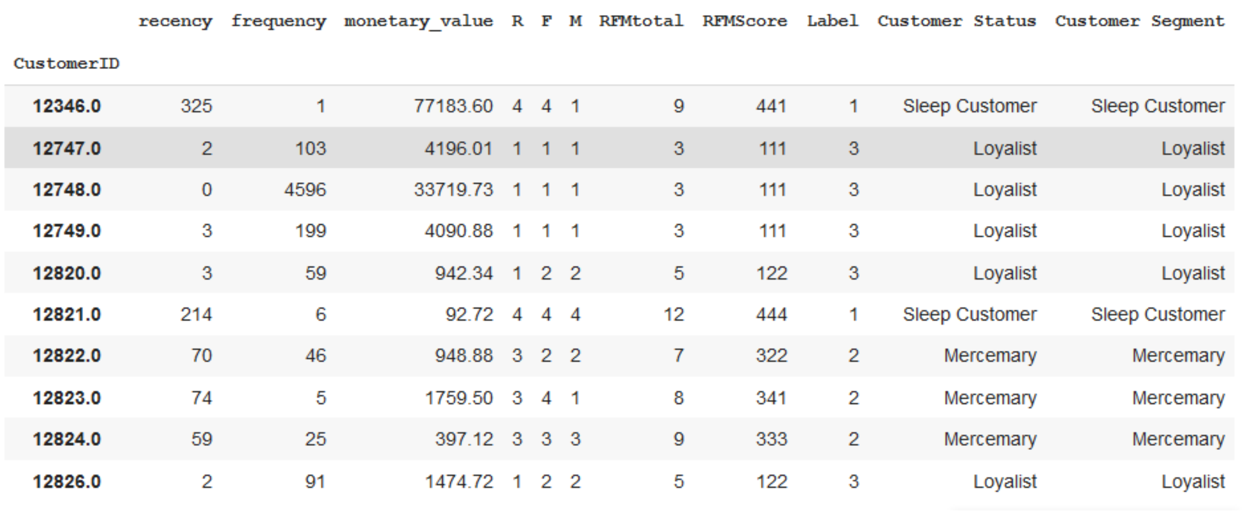
	recency	frequency	monetary_value	R	F	M	RFMtotal	RFMScore	Label
CustomerID									
12346.0	325	1	77183.60	4	4	1	9	441	1
12747.0	2	103	4196.01	1	1	1	3	111	0
12748.0	0	4596	33719.73	1	1	1	3	111	0
12749.0	3	199	4090.88	1	1	1	3	111	0
12820.0	3	59	942.34	1	2	2	5	122	0

- The last we just need to give every customer segment's name: Loyalist, Hostage, Mercenary and Sleep Customer. To give the name, you can use the syntax below.



```
def getStatus(x):
  if x == 3:
    return "Loyalist"
  elif x == 0:
    return "Hostage"
  elif x == 2:
    return "Mercemary"
  else:
    return "Sleep Customer"

rfm4['Customer Segment'] = rfm4['Label'].apply(getStatus)
rfm4.head(10)
```

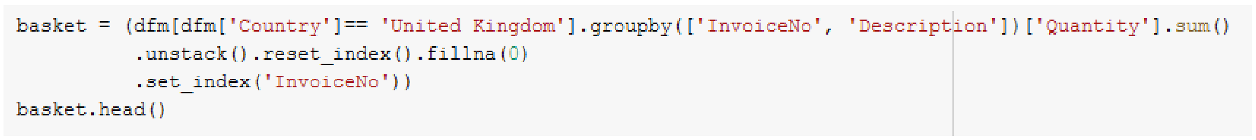
- The picture below is the customer segmentation result.



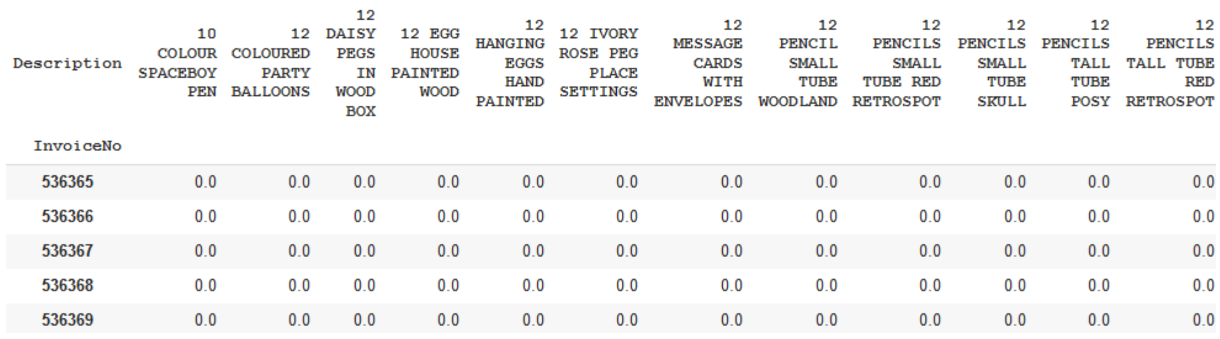
	recency	frequency	monetary_value	R	F	M	RFMtotal	RFMScore	Label	Customer Status	Customer Segment
CustomerID											
12346.0	325	1	77183.60	4	4	1	9	441	1	Sleep Customer	Sleep Customer
12747.0	2	103	4196.01	1	1	1	3	111	3	Loyalist	Loyalist
12748.0	0	4596	33719.73	1	1	1	3	111	3	Loyalist	Loyalist
12749.0	3	199	4090.88	1	1	1	3	111	3	Loyalist	Loyalist
12820.0	3	59	942.34	1	2	2	5	122	3	Loyalist	Loyalist
12821.0	214	6	92.72	4	4	4	12	444	1	Sleep Customer	Sleep Customer
12822.0	70	46	948.88	3	2	2	7	322	2	Mercemary	Mercemary
12823.0	74	5	1759.50	3	4	1	8	341	2	Mercemary	Mercemary
12824.0	59	25	397.12	3	3	3	9	333	2	Mercemary	Mercemary
12826.0	2	91	1474.72	1	2	2	5	122	3	Loyalist	Loyalist

Market Basket Analysis

- Create the basket data by filter the country (United Kingdom), group by Invoice Number and Description with sum of quantity as its aggregate function. With this process we will get the items transaction for every invoceNo based on all existing items.

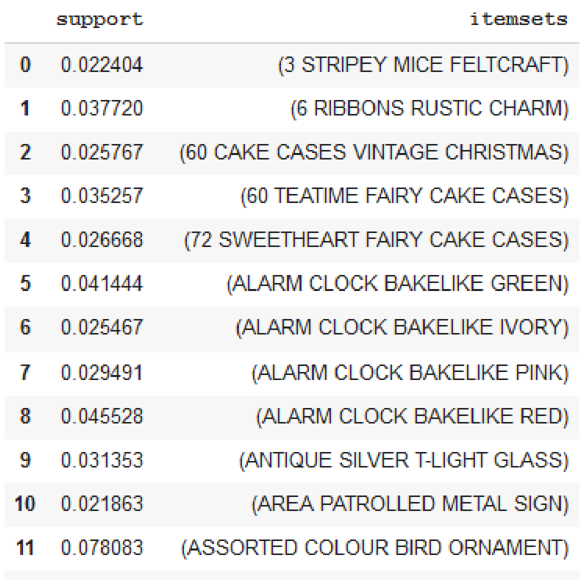


```
basket = (dfm[dfm['Country']== 'United Kingdom'].groupby(['InvoiceNo', 'Description'])['Quantity'].sum()
          .unstack().reset_index().fillna(0)
          .set_index('InvoiceNo'))
basket.head()
```



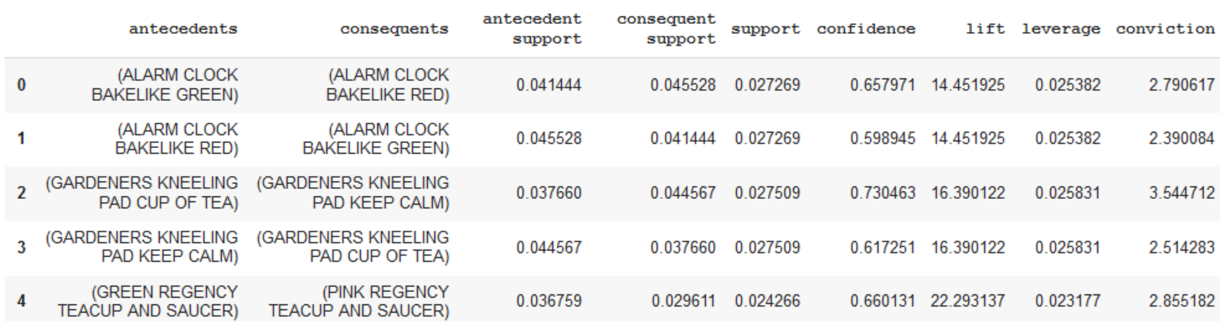
Description	10 COLOUR SPACEBOY PEN	12 COLOURED PARTY BALLOONS	12 DAISY PEGS IN WOOD BOX	12 EGG HOUSE PAINTED WOOD	12 HANGING EGGS HAND PAINTED	12 IVORY ROSE PEG PLACE SETTINGS	12 MESSAGE CARDS WITH ENVELOPES	12 PENCIL SMALL TUBE WOODLAND	12 PENCILS SMALL TUBE RED RETROSPOT	12 PENCILS SMALL TUBE SKULL	12 PENCILS TALL TUBE POSY	12 PENCILS TALL TUBE RED RETROSPOT
InvoiceNo												
536365	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
536366	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
536367	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
536368	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
536369	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0

- Because we don't need the quantity number in every invoiceNo, we create function to change the data. If transaction == 0 set the value as 0, if the value more than 0 set the value as 1.
- To create this model, I used apriori with association rules. First, build your apriori to get the percentage of items are frequently purchased from all transaction. Don't forget to give min_support to filter that data. Below is the result from this process.



	support	itemsets
0	0.022404	(3 STRIPEY MICE FELTCRAFT)
1	0.037720	(6 RIBBONS RUSTIC CHARM)
2	0.025767	(60 CAKE CASES VINTAGE CHRISTMAS)
3	0.035257	(60 TEATIME FAIRY CAKE CASES)
4	0.026668	(72 SWEETHEART FAIRY CAKE CASES)
5	0.041444	(ALARM CLOCK BAKELIKE GREEN)
6	0.025467	(ALARM CLOCK BAKELIKE IVORY)
7	0.029491	(ALARM CLOCK BAKELIKE PINK)
8	0.045528	(ALARM CLOCK BAKELIKE RED)
9	0.031353	(ANTIQUE SILVER T-LIGHT GLASS)
10	0.021863	(AREA PATROLLED METAL SIGN)
11	0.078083	(ASSORTED COLOUR BIRD ORNAMENT)

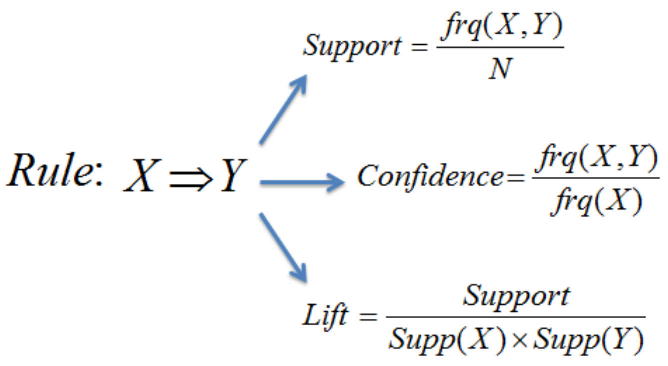
• The last is build the association rules, and below is the result from this process.



	antecedents	consequents	antecedent support	consequent support	support	confidence	lift	leverage	conviction
0	(ALARM CLOCK BAKELIKE GREEN)	(ALARM CLOCK BAKELIKE RED)	0.041444	0.045528	0.027269	0.657971	14.451925	0.025382	2.790617
1	(ALARM CLOCK BAKELIKE RED)	(ALARM CLOCK BAKELIKE GREEN)	0.045528	0.041444	0.027269	0.598945	14.451925	0.025382	2.390084
2	(GARDENERS KNEELING PAD CUP OF TEA)	(GARDENERS KNEELING PAD KEEP CALM)	0.037660	0.044567	0.027509	0.730463	16.390122	0.025831	3.544712
3	(GARDENERS KNEELING PAD KEEP CALM)	(GARDENERS KNEELING PAD CUP OF TEA)	0.044567	0.037660	0.027509	0.617251	16.390122	0.025831	2.514283
4	(GREEN REGENCY TEACUP AND SAUCER)	(PINK REGENCY TEACUP AND SAUCER)	0.036759	0.029611	0.024266	0.660131	22.293137	0.023177	2.855182

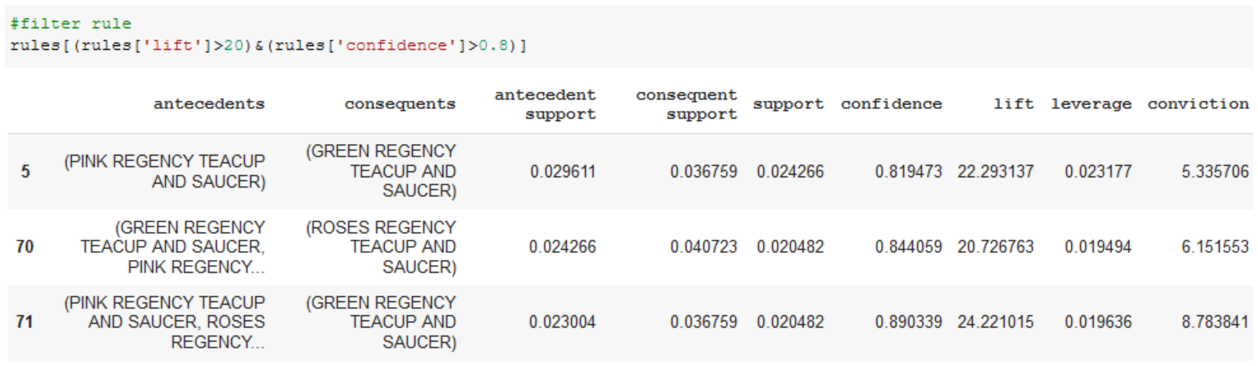
- ***Support*** is the number of transactions that include items in the {A} and {B} parts of the rule as a percentage of the total number of transactions. It is a measure of how frequently the collection of items occur together as a percentage of all transactions.
- ***Confidence*** is the ratio of the number of transactions that include all items in {B} as well as the number of transactions that include all items in {A} to the number of transactions that include all items in {A}.
- ***Lift*** is the ratio of confidence to expected confidence. Expected confidence is the confidence divided by the frequency of B. The Lift tells us how much better a rule is at predicting the result than just assuming the result in the first place. Greater lift values indicate stronger associations.

More a clearer formula, see the following image


$Rule: X \Rightarrow Y$
$Support = \frac{frq(X,Y)}{N}$
$Confidence = \frac{frq(X,Y)}{frq(X)}$
$Lift = \frac{Support}{Supp(X) \times Supp(Y)}$

• Based on that data we know that (index:4) customer who bought "*Green Regency Teacup and Saucer*" also bought "*Pink Regency Teacup and Saucer*". Based on this case, as a seller ***we can decided to put the items at the same shelf to increase the items sales.***

- You can also filter the data to get more specific items purchased together (like confidence and lift)



```
#filter rule
rules[(rules['lift']>20)&(rules['confidence']>0.8)]
```

	antecedents	consequents	antecedent support	consequent support	support	confidence	lift	leverage	conviction
5	(PINK REGENCY TEACUP AND SAUCER)	(GREEN REGENCY TEACUP AND SAUCER)	0.029611	0.036759	0.024266	0.819473	22.293137	0.023177	5.335706
70	(GREEN REGENCY TEACUP AND SAUCER, PINK REGENCY...	(ROSES REGENCY TEACUP AND SAUCER)	0.024266	0.040723	0.020482	0.844059	20.726763	0.019494	6.151553
71	(PINK REGENCY TEACUP AND SAUCER, ROSES REGENCY...	(GREEN REGENCY TEACUP AND SAUCER)	0.023004	0.036759	0.020482	0.890339	24.221015	0.019636	8.783841

Deployment

Deployment is the last process of CRISP-DM framework after evaluate the model. Based on this case we can deploy this model on the website or apps depends on requirement. But we know that our main purpose is how to get the customer segmentation if there is a new customer or what the product that can put at the same self. May be we can build this model as API and use the API inside the website.

Jump to this link for the code:
https://github.com/irwanafandi24/IYKRA_Bootcamp/tree/master/Week_4_Machine_Learning

Conclusion

Customer segmentation and Market Basket Analysis are the method to increase the items sales in the retail store. With customer segmentation we know who is the defector, mercenary, hostage and loyalist customer. Now, we can treat our customer as best as we can. We can give the loyalist customer some discount, extra bonus, and soon to make them stay loyal. We also can give the hostage or mercenary customer good service, discount or other to make the categorize move to loyal. So, there are a lot of benefit if we can mapping our customers. In other hand, Market Basket Analysis is one of the method that can increase our items sales. With this method we can see the behavior of every customer like if they buy noodles they will buy eggs too, or if they buy breads they will buy milk too, etc. Based on this case, we decided to put the couple of items at the same shelf. Sometimes, it can influence the customers to buy the item even though they did not plan to buy it before. And that is the goal, to increase the items sales.