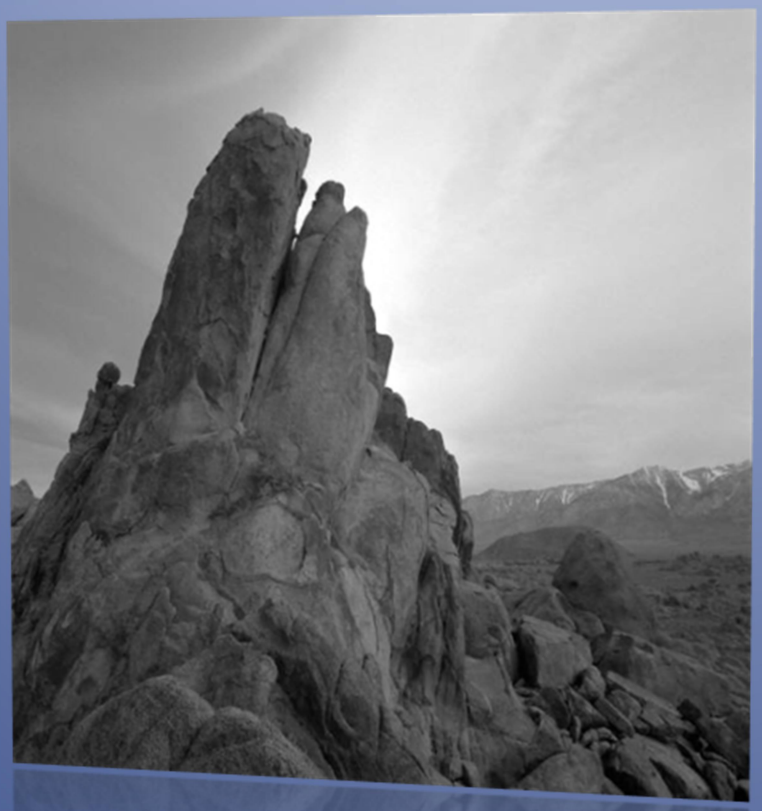


Data mining operations

Data mining semester project 2021

This document is the report of dataming project. It contains the main basic learned concepts dusring the whole semester. It contains the concepts and practical examples for the topics like pre-processing techniques and using the multiple classification algorithms and understanding calculating the performance measures of those classisfication algorithms. For the purpose of performing these all operation two datasets are taken from KAGGEL and links are provided for the whole code which is specialty written by author (Zeeshan Ahmed) is given with links of github

Table of Contents

How will be the flow of this project?

- We have two datasets; choose so that I may cover all the concepts and issues. So both will be used for different purposes.
- we know that for achieving the requirements of this project like graphing techniques along with usage and understanding the usage of those graph's results **we need numeric data.** So for this we will convert columns into numeric data by using python. In last we have select the features (will drop some columns on the basic of two assumptions that either they are unnecessary or there some other strongly co-related columns).
- So 1st of all we will perform some pre-processing on both datasets separately and will try to identify the different types of pre-processing issues and their solution buy using python. While doing pre-processing we will also perform some steps which make data standardized. In the result of pre-processing we will get the two new CSV files, one with non-standardized values and other with standardized values, which will be processed further for rest of operations.
- For the purpose of graph to visualize and understanding of the data using boxplot, histogram, qq-plot, scatter plot and etc, we will use the dataset 1. We will also analyze the effects of using standardized data and non-standardized data.
- For the purpose of Apply machine learning algorithm for finding frequent patterns, classification, clustering we will make use of both pre-processed dataset files.
- In last we will make the confusion matrix which will show the accuracy and cross validation scores, in case of classification algorithms, and other interesting findings which will be gathered by changing the parameters for algorithms.

NOTE: I have used two datasets in this assignment. The reason for choosing different datasets is to cover as many as concepts of data mining.

Dataset 1 (big-mart-sales) source: https://github.com/ZeeWING-Projects/DM-Project/blob/main/RealDataSet/RealDataSet.csv

Dataset 2 (startup-success-prediction) source: https://www.kaggle.com/manishkc06/startup-success-prediction)

Whole code used for this project and other resources: https://github.com/ZeeWING-Projects/DM-Project

Tools : Jupiter note book for running code

Note: For the purpose of results used in this document I have run provided code in different ways, by running some specific parts sometimes whole code at a time. So you might need to un comment some code and comment some code to get those results.

Pre-processing

Dataset 1 (big-mart-sales):

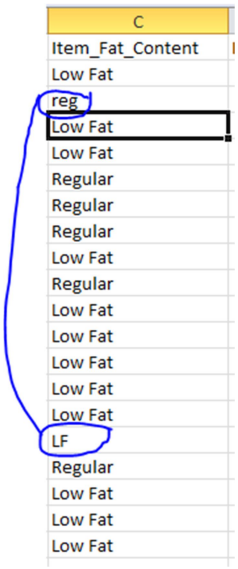
1. Applying some pre-processing steps (Data set 1):

First we need to apply some pre-processing techniques before we process it.

Find Code at : https://github.com/ZeeWING-Projects/DM-Project/blob/main/Preprocessing-Code/Preprocessing-dataset-1.ipynb

1.1 Matching field's values:

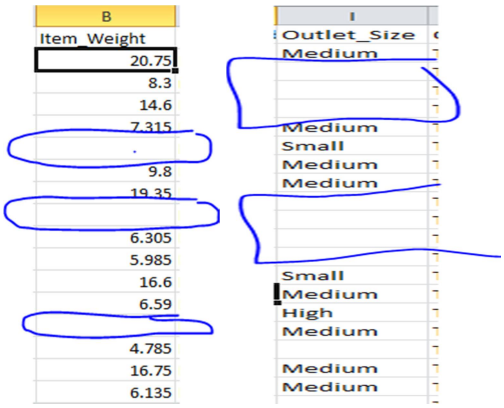
There are few fields in dataset which contain same data with different names, so we need to make the same. For example we have attribute item_Fat_Content which contain two labels, Low Fat and Regular, but for representing this same value "LF" and "reg" are used so we need to remove these shortcuts.



1.2Filling out missing values:

In our dataset we have a bunch of attributes having missing values. And we have to fill them by using well known pre-processing techniques. For example for numeric attribute we have methods like by using median, mean and mode and for ordinal attributes we will use some built in functions of python, like we are using the KNN inputter.

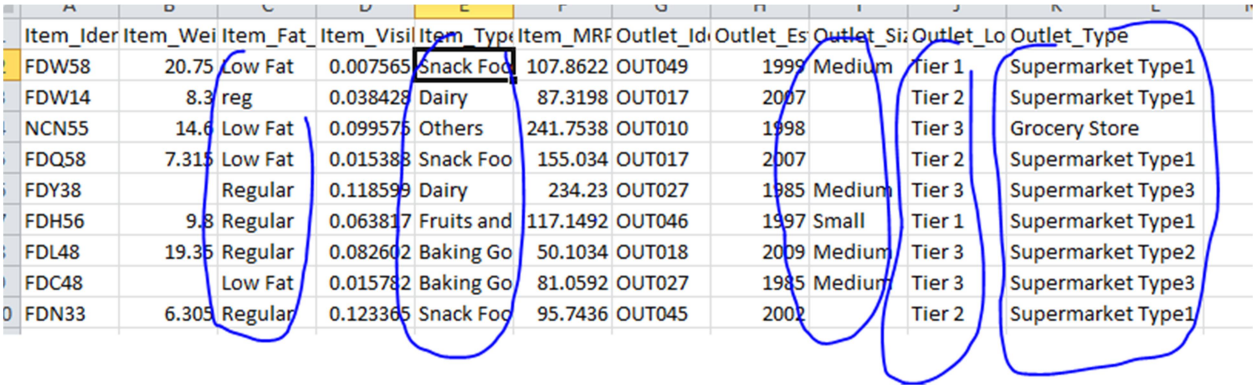
For example we have some attributes with missing values.



B	I
Item_Weight	Outlet_Size
20.75	Medium
8.3	
14.6	
7.315	Medium
.	Small
9.8	Medium
19.35	Medium
6.305	
5.985	
16.6	Small
6.59	Medium
	High
4.785	Medium
16.75	
6.135	Medium
	Medium

1.4 Conversion of string values to numeric values.

We have some values which are in string form so we need to transform them in numeric form. Since we have following values which need to be in numeric form.



Item_Ider	Item_Wei	Item_Fat_	Item_Visi	Item_Type	Item_MRP	Outlet_Id	Outlet_Es	Outlet_Si	Outlet_Lo	Outlet_Type
FDW58	20.75	Low Fat	0.007565	Snack Foo	107.8622	OUT049	1999	Medium	Tier 1	Supermarket Type1
FDW14	8.3	reg	0.038428	Dairy	87.3198	OUT017	2007		Tier 2	Supermarket Type1
NCN55	14.6	Low Fat	0.099575	Others	241.7538	OUT010	1998		Tier 3	Grocery Store
FDQ58	7.315	Low Fat	0.015383	Snack Foo	155.034	OUT017	2007		Tier 2	Supermarket Type1
FDY38		Regular	0.118599	Dairy	234.23	OUT027	1985	Medium	Tier 3	Supermarket Type3
FDH56	9.8	Regular	0.063817	Fruits and	117.1492	OUT046	1997	Small	Tier 1	Supermarket Type1
FDL48	19.35	Regular	0.082602	Baking Go	50.1034	OUT018	2009	Medium	Tier 3	Supermarket Type2
FDC48		Low Fat	0.015782	Baking Go	81.0592	OUT027	1985	Medium	Tier 3	Supermarket Type3
FDN33	6.305	Regular	0.123365	Snack Foo	95.7436	OUT045	2002		Tier 2	Supermarket Type1

As you can see clearly that these are some columns which are like normal values but some are showing that those can be used as class label. So we are assuming that feature named **Outlet_Type will be used for classification purpose**. So it will be translated accordingly function.

1.5 Conversion of normal values to standardized values.

We have some values which are stated in 100s unit and some are 1s unit. What I meant is that as we have following values.

	A	B	C	D	E	F	G	H	I	J	K	L
1	Item_Iden	Item_Wei	Item_Fat_	Item_Visi	Item_Type	Item_MRF	Outlet_Id	Outlet_Es	Outlet_Si	Outlet_Lo	Outlet_Type	
2	FDW58	20.75	Low Fat	0.007565	Snack Foo	107.8622	OUT049	1999	Medium	Tier 1	Supermarket Type1	
3	FDW14	8.3	reg	0.038428	Dairy	87.3198	OUT017	2007		Tier 2	Supermarket Type1	
4	NCN55	14.6	Low Fat	0.099575	Others	241.7538	OUT010	1998		Tier 3	Grocery Store	
5	FDQ58	7.315	Low Fat	0.015388	Snack Foo	155.034	OUT017	2007		Tier 2	Supermarket Type1	
6	FDY38		Regular	0.118599	Dairy	234.23	OUT027	1985	Medium	Tier 3	Supermarket Type3	
7	FDH56	9.8	Regular	0.063817	Fruits and	117.1492	OUT046	1997	Small	Tier 1	Supermarket Type1	
8	FDL48	19.35	Regular	0.082602	Baking Go	50.1034	OUT018	2009	Medium	Tier 3	Supermarket Type2	
9	FDC48		Low Fat	0.015782	Baking Go	81.0592	OUT027	1985	Medium	Tier 3	Supermarket Type3	
10	FDN33	6.305	Regular	0.123365	Snack Foo	95.7436	OUT045	2002		Tier 2	Supermarket Type1	
11	FDA36	5.985	Low Fat	0.005698	Baking Go	186.8924	OUT017	2007		Tier 2	Supermarket Type1	
12	FDT44	16.6	Low Fat	0.103569	Fruits and	118.3466	OUT017	2007		Tier 2	Supermarket Type1	

As highlighted values are those one which need to be standardized because this will affect the co-relation graph, which will be drawn later on.

1.5Implementation of above stated issues

So we have a real dataset having some missing values and some other issues like **conversion of ordinal values to numeric form** so we need to perform above stated steps of pre-processing.

Before filling missing values

```
#Before filling missing values.
data.head(10)
```

	Item_Identifier	Item_Weight	Item_Fat_Content	Item_Visibility	Item_Type	Item_MRP	Outlet_Identifier	Outlet_Establishment_Year	Outlet_Size	Outlet_Locatio
0	FDW58	20.750	Low Fat	0.007565	Snack Foods	107.8622	OUT049	1999	Medium	
1	FDW14	8.300	Regular	0.038428	Dairy	87.3198	OUT017	2007	NaN	
2	NCN55	14.600	Low Fat	0.099575	Others	241.7538	OUT010	1998	NaN	
3	FDQ58	7.315	Low Fat	0.015388	Snack Foods	155.0340	OUT017	2007	NaN	
4	FDY38	NaN	Regular	0.118599	Dairy	234.2300	OUT027	1985	Medium	
5	FDH56	9.800	Regular	0.063817	Fruits and Vegetables	117.1492	OUT046	1997	Small	
6	FDL48	19.350	Regular	0.082602	Baking Goods	50.1034	OUT018	2009	Medium	
7	FDC48	NaN	Low Fat	0.015782	Baking Goods	81.0592	OUT027	1985	Medium	
8	FDN33	6.305	Regular	0.123365	Snack Foods	95.7436	OUT045	2002	NaN	

After filling missing values

```
#After filling out missing values
data.head(20)
```

	Item_Identifier	Item_Weight	Item_Fat_Content	Item_Visibility	Item_Type	Item_MRP	Outlet_Identifier	Outlet_Establishment_Year	Outlet_Size	Outlet_Locati
0	FDW58	20.750	Low Fat	0.007565	Snack Foods	107.8622	OUT049	1999.0	Medium	
1	FDW14	8.300	Regular	0.038428	Dairy	87.3198	OUT017	2007.0	Small	
2	NCN55	14.600	Low Fat	0.099575	Others	241.7538	OUT010	1998.0	Medium	
3	FDQ58	7.315	Low Fat	0.015388	Snack Foods	155.0340	OUT017	2007.0	Medium	
4	FDY38	14.618	Regular	0.118599	Dairy	234.2300	OUT027	1985.0	Medium	
5	FDH56	9.800	Regular	0.063817	Fruits and Vegetables	117.1492	OUT046	1997.0	Small	
6	FDL48	19.350	Regular	0.082602	Baking Goods	50.1034	OUT018	2009.0	Medium	
7	FDC48	10.225	Low Fat	0.015782	Baking Goods	81.0592	OUT027	1985.0	Medium	
8	FDN33	6.305	Regular	0.123365	Snack Foods	95.7436	OUT045	2002.0	Medium	

Before converting string values to numeric form

Item_Weight	Item_Fat_Content	Item_Visibility	Item_Type	Item_MRP	Outlet_Identifier	Outlet_Establishment_Year	Outlet_Size	Outlet_Location_Type	Outlet_Type
20.750	Low Fat	0.007565	Snack Foods	107.8622	OUT049	1999.0	Medium	Tier 1	Supermarket Type1
8.300	Regular	0.038428	Dairy	87.3198	OUT017	2007.0	Medium	Tier 2	Supermarket Type1
14.600	Low Fat	0.099575	Others	241.7538	OUT010	1998.0	Medium	Tier 3	Grocery Store
7.315	Low Fat	0.015388	Snack Foods	155.0340	OUT017	2007.0	Medium	Tier 2	Supermarket Type1
14.618	Regular	0.118599	Dairy	234.2300	OUT027	1985.0	Medium	Tier 3	Supermarket Type3
9.800	Regular	0.063817	Fruits and Vegetables	117.1492	OUT046	1997.0	Small	Tier 1	Supermarket Type1
19.350	Regular	0.082602	Baking Goods	50.1034	OUT018	2009.0	Medium	Tier 3	Supermarket Type2

After converting string values to numeric form

```
data.head(10)
```

Item_Weight	Item_Fat_Content	Item_Visibility	Item_Type	Item_MRP	Outlet_Identifier	Outlet_Establishment_Year	Outlet_Size	Outlet_Location_Type	Outlet_Type
20.750	1.0	0.007565	13.0	107.8622	OUT049	1999.0	1.0	0.0	1.0
8.300	0.0	0.038428	4.0	87.3198	OUT017	2007.0	1.0	1.0	1.0
14.600	1.0	0.099575	11.0	241.7538	OUT010	1998.0	1.0	2.0	0.0
7.315	1.0	0.015388	13.0	155.0340	OUT017	2007.0	1.0	1.0	1.0
14.618	0.0	0.118599	4.0	234.2300	OUT027	1985.0	1.0	2.0	3.0
9.800	0.0	0.063817	6.0	117.1492	OUT046	1997.0	2.0	0.0	1.0
19.350	0.0	0.082602	0.0	50.1034	OUT018	2009.0	1.0	2.0	2.0
10.225	1.0	0.015782	0.0	81.0592	OUT027	1985.0	1.0	2.0	3.0
6.305	0.0	0.123365	13.0	95.7436	OUT045	2002.0	0.0	1.0	1.0
5.985	1.0	0.005698	0.0	186.8924	OUT017	2007.0	1.0	1.0	1.0

Before standardization of values

Since we don't standardize the labels:
(https://www.google.com/search?q=should+we+standarize+the+lablel+values&oq=should+we+standarize+the+lablel+values+&aqs=chrome..69i57j33i10i160l4.16858j0j4&sourceid=chrome&ie=UTF-8)

Item_Weight	Item_Fat_Content	Item_Visibility	Item_Type	Item_MRP	Outlet_Identifier	Outlet_Establishment_Year	Outlet_Size	Outlet_Location_Type	Outlet_Type
20.750	1.0	0.007565	13.0	107.8622	OUT049	1999.0	1.0	0.0	1.0
8.300	0.0	0.038428	4.0	87.3198	OUT017	2007.0	2.0	1.0	1.0
14.600	1.0	0.099575	11.0	241.7538	OUT010	1998.0	2.0	2.0	0.0
7.315	1.0	0.015388	13.0	155.0340	OUT017	2007.0	2.0	1.0	1.0
14.618	0.0	0.118599	4.0	234.2300	OUT027	1985.0	1.0	2.0	3.0
9.800	0.0	0.063817	6.0	117.1492	OUT046	1997.0	2.0	0.0	1.0
19.350	0.0	0.082602	0.0	50.1034	OUT018	2009.0	1.0	2.0	2.0
10.225	1.0	0.015782	0.0	81.0592	OUT027	1985.0	1.0	2.0	3.0
6.305	0.0	0.123365	13.0	95.7436	OUT045	2002.0	2.0	1.0	1.0
5.985	1.0	0.005698	0.0	186.8924	OUT017	2007.0	1.0	1.0	1.0

After standardization of values

Item_Weight	Item_Fat_Content	Item_Visibility	Item_Type	Item_MRP	Outlet_Identifier	Outlet_Establishment_Year	Outlet_Size	Outlet_Location_Type	Outlet_Type
1.871702	1.0	-1.134077	13.0	-0.536555	OUT049	1999.0	1.0	0.0	1.0
-1.002867	0.0	-0.531850	4.0	-0.868937	OUT017	2007.0	2.0	1.0	1.0
0.451735	1.0	0.661316	11.0	1.629848	OUT010	1998.0	2.0	2.0	0.0
-1.230292	1.0	-0.981416	13.0	0.226697	OUT017	2007.0	2.0	1.0	1.0
0.455891	0.0	1.032540	4.0	1.508110	OUT027	1985.0	1.0	2.0	3.0
-0.656533	0.0	-0.036424	6.0	-0.386289	OUT046	1997.0	2.0	0.0	1.0
1.548458	0.0	0.330115	0.0	-1.471108	OUT018	2009.0	1.0	2.0	2.0
-0.558405	1.0	-0.973726	0.0	-0.970235	OUT027	1985.0	1.0	2.0	3.0
-1.463490	0.0	1.125541	13.0	-0.732638	OUT045	2002.0	2.0	1.0	1.0
-1.537375	1.0	-1.170496	0.0	0.742175	OUT017	2007.0	1.0	1.0	1.0

Now making selection of features

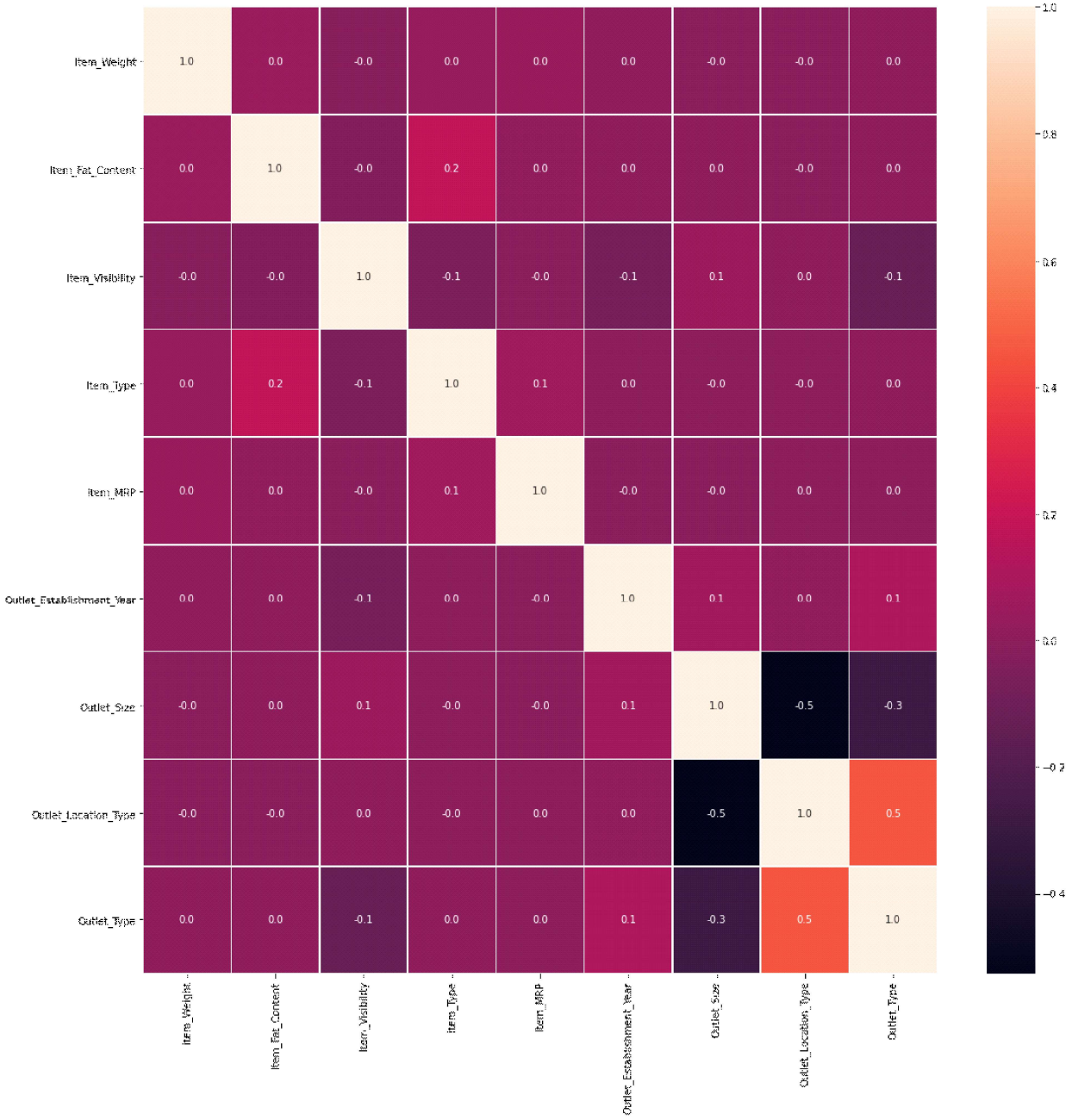
Columns before selection

```
#Columns before making selection.
data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 5681 entries, 0 to 5680
Data columns (total 11 columns):
 #   Column                     Non-Null Count  Dtype  
---  ------                     --------------  -----  
 0   Item_Identifier            5681 non-null   object 
 1   Item_Weight                5681 non-null   float64
 2   Item_Fat_Content           5615 non-null   float64
 3   Item_Visibility            5681 non-null   float64
 4   Item_Type                  5681 non-null   float64
 5   Item_MRP                   5681 non-null   float64
 6   Outlet_Identifier          5681 non-null   object 
 7   Outlet_Establishment_Year  5681 non-null   float64
 8   Outlet_Size                5681 non-null   float64
 9   Outlet_Location_Type       5681 non-null   float64
 10  Outlet_Type                5681 non-null   float64
dtypes: float64(9), object(2)
memory usage: 488.3+ KB
```

Columns after selection

We will drop two columns which are just showing the IDs.



So from this we can say that there is no column is strongly related with any other so we don't need to remove any column. Because these will perform important role In classification.

Dataset (standardized values) : https://github.com/ZeeWING-Projects/DM-Project/blob/main/Dataset-1%20Pre-processed/Dataset_01_standarized_.csv

Dataset (non-standardized values) : https://github.com/ZeeWING-Projects/DM-Project/blob/main/Dataset-1%20Pre-processed/Dataset_01_non_standarized_.csv

Dataset 2 (start-up-success-pridiction):

2. Applying some pre-processing steps (Data set 2):

First we need to apply some pre-processing techniques before we process it.

Find whole code at : https://github.com/ZeeWING-Projects/DM-Project/blob/main/Preprocessing-Code/Preprocessing-dataset-2.ipynb

2.2Filling out missing values:

In our dataset we have a bunch of attributes having missing values. And we have to fill them by using well known pre-processing techniques. For example for numeric attribute we have methods like by using median, mean and mode and for ordinal attributes we will use some built in functions of python, like we are using the KNN inputter.

For example we have some attributes with missing values. As there 49 columns so I am just showing by using this form of result.

```
city                        0
Unnamed: 6                493
name                        0
labels                      0
founded_at                  0
closed_at                 588
first_funding_at            0
last_funding_at             0
age_first_funding_year      0
age_last_funding_year       0
age_first_milestone_year  152
age_last_milestone_year   152
relationships               0
funding_rounds              0
funding_total_usd           0
milestones                  0
state_code.1                1
is_CA                       0
is_NY                       0
is_MA                       0
```

2.2 Conversion of string values to numeric values.

We have some values which are in string form so we need to transform them in numeric form. Since we have following values which need to be in numeric form.

```
Data columns (total 49 columns):
 #   Column                   Non-Null Count  Dtype
---  ------                   --------------  -----
 0   Unnamed: 0               923 non-null    int64
 1   state_code               923 non-null    object
 2   latitude                 923 non-null    float64
 3   longitude                923 non-null    float64
 4   zip_code                 923 non-null    object
 5   id                       923 non-null    object
 6   city                     923 non-null    object
 7   Unnamed: 6               430 non-null    object
 8   name                     923 non-null    object
 9   labels                   923 non-null    int64
 10  founded_at               923 non-null    object
 11  closed_at                335 non-null    object
 12  first_funding_at         923 non-null    object
 13  last_funding_at          923 non-null    object
 14  age_first_funding_year   923 non-null    float64
 15  age_last_funding_year    923 non-null    float64
 16  age_first_milestone_year 771 non-null    float64
 17  age_last_milestone_year  771 non-null    float64
 18  relationships            923 non-null    int64
 19  funding_rounds           923 non-null    int64
 20  funding_total_usd        923 non-null    int64
 21  milestones               923 non-null    int64
 22  state_code.1             922 non-null    object
 23  is_CA                    923 non-null    int64
 24  is_NY                    923 non-null    int64
 25  is_MA                    923 non-null    int64
 26  is_TX                    923 non-null    int64
 27  is_otherstate            923 non-null    int64
 28  category_code            923 non-null    object
 29  is_software              923 non-null    int64
 30  is_web                   923 non-null    int64
 31  is_mobile                923 non-null    int64
 32  is_enterprise            923 non-null    int64
 33  is_advertising           923 non-null    int64
 34  is_gamesvideo            923 non-null    int64
 35  is_ecommerce             923 non-null    int64
 36  is_biotech               923 non-null    int64
 37  is_consulting            923 non-null    int64
 38  is_othercategory         923 non-null    int64
 39  object_id                923 non-null    object
 40  has_VC                   923 non-null    int64
 41  has_angel                923 non-null    int64
 42  has_roundA               923 non-null    int64
 43  has_roundB               923 non-null    int64
```

1.5 Conversion of normal values to standardized values.

We have some values which are stated in 100s unit and some are 1s unit. What I meant is that as we have following values.

S	T	U	V
relationsh	funding_r	funding_t	milestone
3	3	375000	3
9	4	40100000	1
5	1	2600000	2
5	3	40000000	1
2	2	1300000	1
3	1	7500000	1
6	3	26000000	2
25	3	34100000	3
13	3	9650000	4
14	3	5750000	4
22	3	27500000	3
8	5	10400000	2
0	1	350000	0
15	3	9950000	3
12	5	10700000	1
0	1	200000	0
8	5	49000000	1

AU
avg_participants
1
4.75
4
3.3333
1
3
1.6667
3.5
4
1
1
1.75
1
2.3333

latitude	longitude	zip_code
42.35888	-71.0568	92101
37.23892	-121.974	95032
32.90105	-117.193	92121
37.32031	-122.05	95014
37.77928	-122.419	94105
37.40691	-122.09	94043
37.39156	-122.07	94041
38.05711	-122.514	94901
42.71221	-73.2036	1267
37.42724	-122.146	94306
37.44299	-122.162	94025
37.45299	-122.185	94025
38.24147	-85.7245	40204
40.70276	-73.9867	11201
39.74627	-104.991	80202

As highlighted values are those one which need to be standardized because this will affect the co-relation graph, which will be drawn later on. And we will have some other columns aswell which will be in numeric form after conversion.

1.5Implementation of above stated issues

So we have a real dataset having some missing values and some other issues like **conversion of ordinal values to numeric form** so we need to perform above stated steps of pre-processing.

Before filling missing values

```
#Before filling null values.
print(data.isnull().sum())
```

```
Unnamed: 0                 0
state_code                 0
latitude                   0
longitude                  0
zip_code                   0
id                         0
city                       0
Unnamed: 6               493
name                       0
labels                     0
founded_at                 0
closed_at                588
first_funding_at           0
last_funding_at            0
age_first_funding_year     0
age_last_funding_year      0
age_first_milestone_year   152
age_last_milestone_year    152
relationships              0
funding_rounds             0
funding_total_usd          0
milestones                 0
```

After filling missing values

```
Unnamed: 0                 0
state_code                 0
latitude                   0
longitude                  0
zip_code                   0
id                         0
city                       0
Unnamed: 6               493
name                       0
labels                     0
founded_at                 0
closed_at                588
first_funding_at           0
last_funding_at            0
age_first_funding_year     0
age_last_funding_year      0
age_first_milestone_year   0
age_last_milestone_year    0
relationships              0
funding_rounds             0
funding_total_usd          0
milestones                 0
state_code.1               1
is_CA                      0
is_NY                      0
```

Since you can observe un named: 6 and closed_at are showing still null values. Reason is that I have used KNN Inputter which works for numbers. So we will try an other method for this.

After filling missing values (after 2nd try)

```
Unnamed: 0                0
state_code                0
latitude                  0
longitude                 0
zip_code                  0
id                        0
city                      0
Unnamed_6                 0
name                      0
labels                    0
founded_at                0
closed_at                 0
first_funding_at          0
last_funding_at           0
age_first_funding_year    0
age_last_funding_year     0
age_first_milestone_year  0
age_last_milestone_year   0
relationships             0
funding_rounds            0
funding_total_usd         0
milestones                0
state_code_1              0
is_CA                     0
is_NY                     0
is_MA                     0
is_TX                     0
is_otherstate             0
category_code             0
is_software               0
is_web                    0
is_mobile                 0
```

Now we have successfully removed all null values.

Before converting string values to numeric form

```
Data columns (total 49 columns):
```

```
 #   Column                    Non-Null Count  Dtype
---  ------                    --------------  -----
 0   Unnamed: 0                923 non-null    float64
 1   state_code                923 non-null    object
 2   latitude                  923 non-null    float64
 3   longitude                 923 non-null    float64
 4   zip_code                  923 non-null    object
 5   id                        923 non-null    object
 6   city                      923 non-null    object
 7   Unnamed_6                 923 non-null    object
 8   name                      923 non-null    object
 9   labels                    923 non-null    float64
 10  founded_at                923 non-null    object
 11  closed_at                 923 non-null    object
 12  first_funding_at          923 non-null    object
 13  last_funding_at           923 non-null    object
 14  age_first_funding_year    923 non-null    float64
 15  age_last_funding_year     923 non-null    float64
 16  age_first_milestone_year  923 non-null    float64
 17  age_last_milestone_year   923 non-null    float64
 18  relationships             923 non-null    float64
 19  funding_rounds            923 non-null    float64
 20  funding_total_usd         923 non-null    float64
 21  milestones                923 non-null    float64
 22  state_code_1              923 non-null    object
 23  is_CA                     923 non-null    float64
 24  is_NY                     923 non-null    float64
 25  is_MA                     923 non-null    float64
 26  is_TX                     923 non-null    float64
 27  is_otherstate             923 non-null    float64
 28  category_code             923 non-null    object
 29  is_software               923 non-null    float64
 30  is_web                    923 non-null    float64
 31  is_mobile                 923 non-null    float64
 32  is_enterprise             923 non-null    float64
 33  is_advertising            923 non-null    float64
 34  is_gamesvideo             923 non-null    float64
 35  is_ecommerce              923 non-null    float64
 36  is_biotech                923 non-null    float64
 37  is_consulting             923 non-null    float64
 38  is_othercategory          923 non-null    float64
 39  object_id                 923 non-null    object
 40  has_VC                    923 non-null    float64
 41  has_angel                 923 non-null    float64
 42  has_roundA                923 non-null    float64
 43  has_roundB                923 non-null    float64
 44  has_roundC                923 non-null    float64
 45  has_roundD                923 non-null    float64
 46  avg_participants          923 non-null    float64
 47  is_top500                 923 non-null    float64
 48  status                    923 non-null    object
```

After converting string values to numeric form

```
 #   Column                    Non-Null Count  Dtype
---  ------                    --------------  -----
 0   Unnamed: 0                923 non-null    float64
 1   state_code                923 non-null    float64
 2   latitude                  923 non-null    float64
 3   longitude                 923 non-null    float64
 4   zip_code                  923 non-null    float64
 5   id                        923 non-null    float64
 6   city                      923 non-null    float64
 7   Unnamed_6                 923 non-null    float64
 8   name                      923 non-null    float64
 9   labels                    923 non-null    float64
 10  founded_at                923 non-null    float64
 11  closed_at                 923 non-null    float64
 12  first_funding_at          923 non-null    float64
 13  last_funding_at           923 non-null    float64
 14  age_first_funding_year    923 non-null    float64
 15  age_last_funding_year     923 non-null    float64
 16  age_first_milestone_year  923 non-null    float64
 17  age_last_milestone_year   923 non-null    float64
 18  relationships             923 non-null    float64
 19  funding_rounds            923 non-null    float64
 20  funding_total_usd         923 non-null    float64
 21  milestones                923 non-null    float64
 22  state_code_1              923 non-null    float64
 23  is_CA                     923 non-null    float64
 24  is_NY                     923 non-null    float64
 25  is_MA                     923 non-null    float64
 26  is_TX                     923 non-null    float64
 27  is_otherstate             923 non-null    float64
 28  category_code             923 non-null    float64
 29  is_software               923 non-null    float64
 30  is_web                    923 non-null    float64
 31  is_mobile                 923 non-null    float64
 32  is_enterprise             923 non-null    float64
 33  is_advertising            923 non-null    float64
 34  is_gamesvideo             923 non-null    float64
 35  is_ecommerce              923 non-null    float64
 36  is_biotech                923 non-null    float64
 37  is_consulting             923 non-null    float64
 38  is_othercategory          923 non-null    float64
 39  object_id                 923 non-null    float64
 40  has_VC                    923 non-null    float64
 41  has_angel                 923 non-null    float64
 42  has_roundA                923 non-null    float64
 43  has_roundB                923 non-null    float64
 44  has_roundC                923 non-null    float64
 45  has_roundD                923 non-null    float64
 46  avg_participants          923 non-null    float64
 47  is_top500                 923 non-null    float64
 48  is_acquired               923 non-null    int64
```

Now we have successfully converted the string value to numeric form.

Before standardization of values

Since we don't standardize the labels:
(https://www.google.com/search?q=should+we+standarize+the+lablel+values&oq=should+we+standarize+the+lablel+values+&aqs=chrome..69i57j33i10i160l4.16858j0j4&sourceid=chrome&ie=UTF-8)

	Unnamed: 0	state_code	latitude	longitude	zip_code	id	city	Unnamed_6	name	labels	...	object_id	has_VC	has_angel	has_roundA	has_roundl
0	1005.0	2.0	42.358880	-71.056820	250.0	811.0	173.0	49.0	75.0	1.0	...	811.0	0.0	1.0	0.0	0.
1	204.0	2.0	37.238916	-121.973718	336.0	170.0	108.0	138.0	781.0	1.0	...	170.0	1.0	0.0	0.0	1.
2	1001.0	2.0	32.901049	-117.192656	251.0	807.0	173.0	187.0	585.0	1.0	...	807.0	0.0	0.0	1.0	0.
3	738.0	2.0	37.320309	-122.050040	333.0	592.0	55.0	57.0	712.0	1.0	...	592.0	0.0	0.0	0.0	1.
4	1002.0	2.0	37.779281	-122.419236	295.0	808.0	174.0	193.0	351.0	0.0	...	808.0	1.0	1.0	0.0	0.

After standardization of values

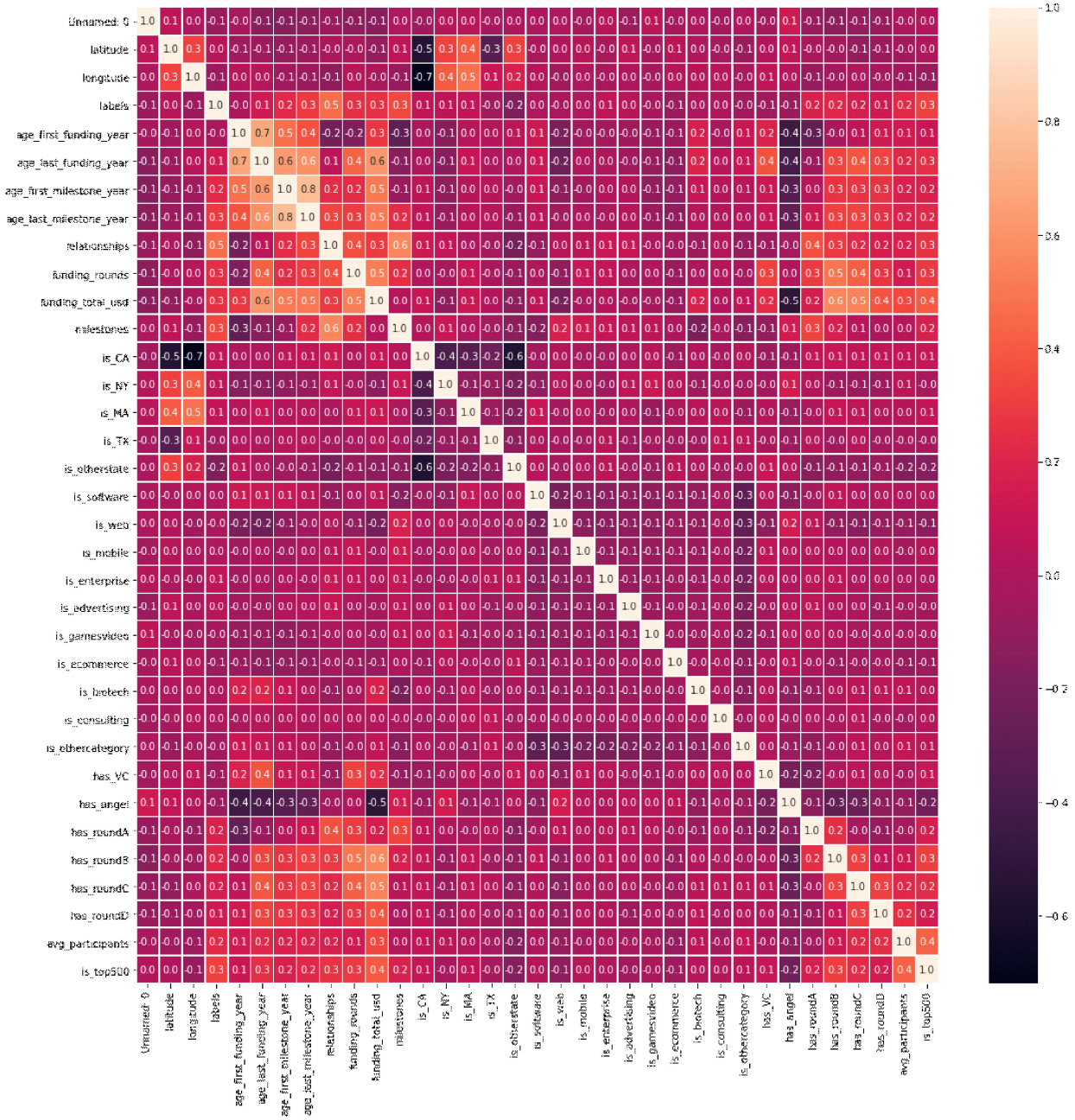
	Unnamed: 0	state_code	latitude	longitude	zip_code	id	city	Unnamed_6	name	labels	...	object_id	has_VC	has_angel	has_rou
0	1.297828	-0.777525	1.027268	1.451271	0.327768	1.317782	0.801958	-1.253114	-1.449539	0.738961	...	1.317782	-0.695646	1.71104	-1.016
1	-1.104657	-0.777525	-0.341900	-0.823630	1.067640	-1.091815	-0.293272	0.035596	1.203933	0.738961	...	-1.091815	1.437514	-0.58444	-1.016
2	1.285831	-0.777525	-1.501922	-0.610018	0.336371	1.302746	0.801958	0.745110	0.467275	0.738961	...	1.302746	-0.695646	-0.58444	0.983
3	0.497000	-0.777525	-0.320134	-0.827040	1.041831	0.494534	-1.186306	-1.137275	0.944599	0.738961	...	0.494534	-0.695646	-0.58444	-1.016
4	1.288830	-0.777525	-0.197397	-0.843535	0.714911	1.306505	0.818808	0.831990	-0.412205	-1.353251	...	1.306505	1.437514	1.71104	-1.016

Now making selection of features

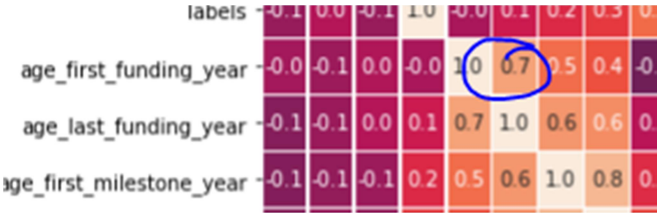
Columns before selection

So we have two reasons to drop a column from dataset.
Either that is un necessary like it may be the id of some column and other reason can be that there exist some other columns which are highly co-related to it, due to that we can remove all and can keep only one column.

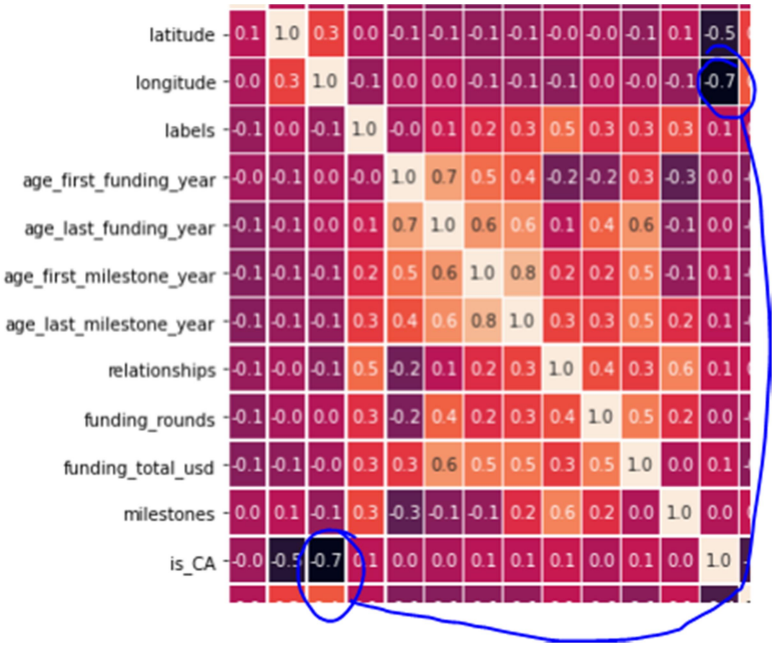
Here we have that graph of co-relation.



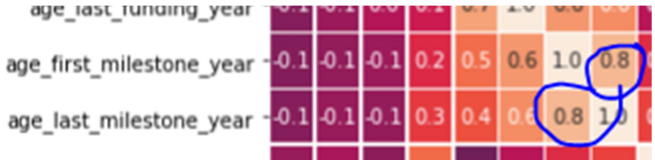
we have summed that for strong relation threshold is 0.7>= so we have will have few columns which need to be remove due to strong relation. Following will be removed.



- See there is strong co-relation between age_first_funding_year and last_funding_year so we need to remove either of them. I am removing first_funding_year.



- See there is strong co-relation between longitude and is_CA so we need to remove either of them. I am removing longitude.



- See there is strong co-relation between age_first_milestone_year and age_last_milestone_ so we need to remove either of them. I am choosing age_last_milestone_
-

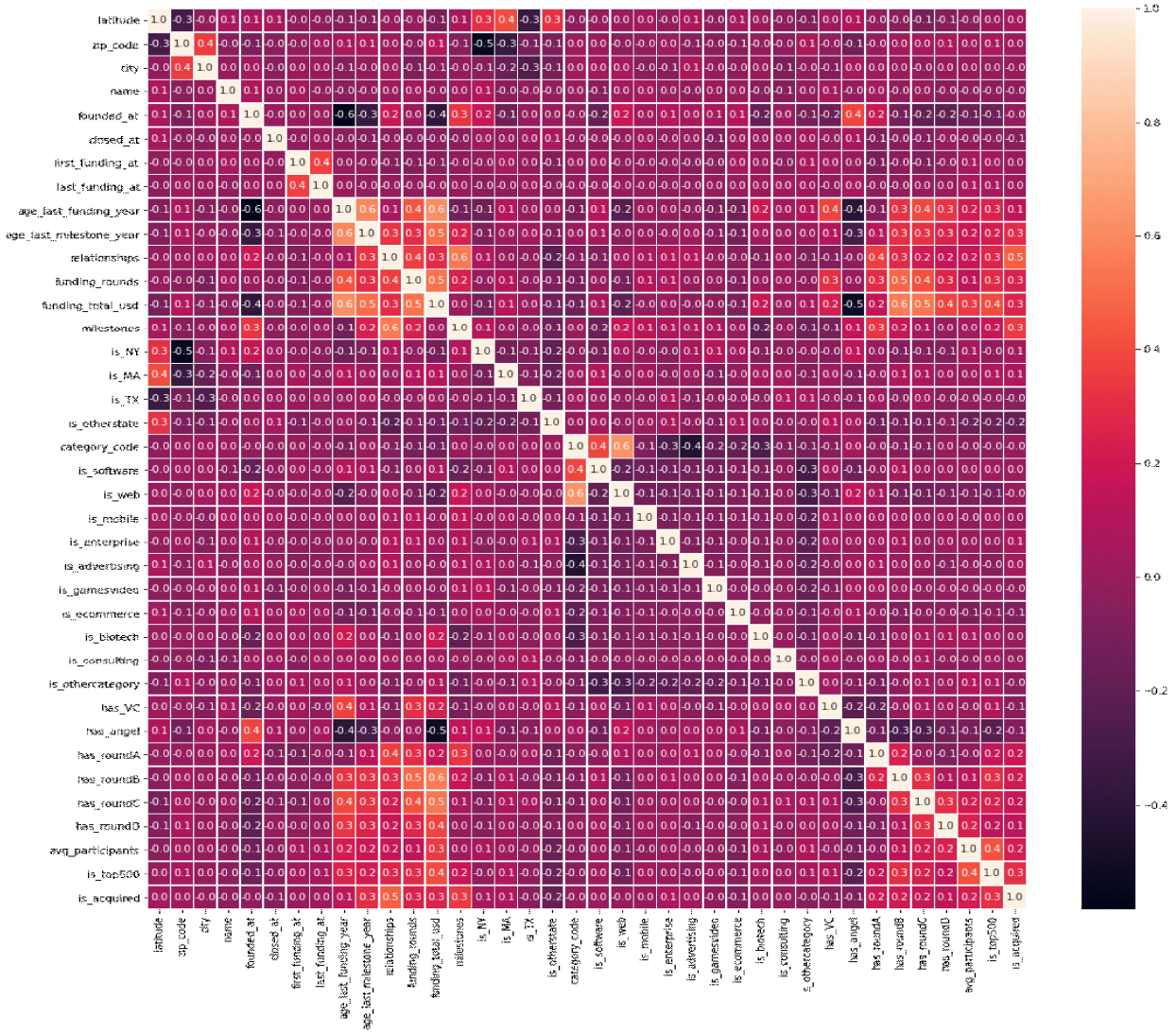
Now we will again remove

• labels

• state_code

• is_CA

After removing these we have following graph



So we are just stopping here but if we want we can remove further features as well, because there are still few reduntant features, but they will not affect the accuracy of our results.

Now we have successfully made a selection of features.

Dataset (standardized values): https://github.com/ZeeWING-Projects/DM-Project/blob/main/Dataset-2%20Pre-processed/Dataset_02_standarized_.csv

Dataset (non-standardized values): https://github.com/ZeeWING-Projects/DM-Project/blob/main/Dataset-2%20Pre-processed/Dataset_02_non_standarized_.csv

Data visualization

NOTE: For this purpose I am using dataset 1. Reason for this is the meaning of features required for understanding of data visualization and scattered plot.

For understating the data graphically we can use different types of graphs. For this dataset analysis we will use some well know graphs to analyze few features of dataset.

Box-Plot

Def: A Box Plot is also known as Whisker plot is created to display the summary of the set of data values having properties like minimum, first quartile, median, third quartile and maximum. In the box plot, a box is created from the first quartile to the third quartile; a vertical line is also there which goes through the box at the median. Here x-axis denotes the data to be plotted while the y-axis shows the frequency distribution. (geeksforgeeks). So this is used for quick summary of data. we have following points to observe by using this box-plot.

- ✓ Detect outlier values.
- ✓ Mean tendency of values.
- ✓ Symmetry of data.

Assume for example we are plotting item_weight of different types of items (item_type)

For this we need item weights of each individual item_type. we have following types of unique items

NOTE: We are using non-standardized dataset (the one which is already pre-processed see above section)

but as we have converted the all values in numeric form then now we need to match columns so that we be able to get the names against numbers.

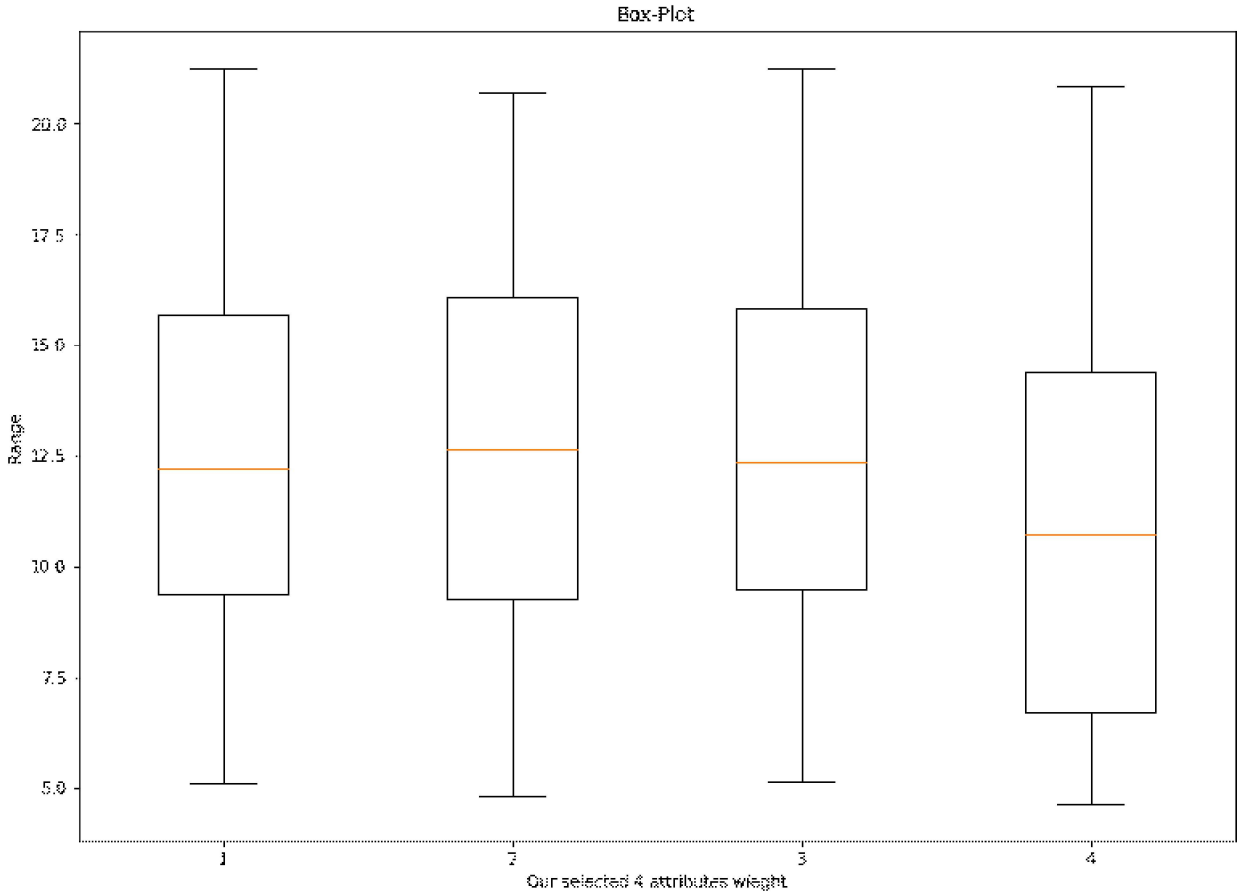
	Item_Weight	Item_Fat_Content	Item_Visibility	Item_Type	I
0	1.871702	1.0	-1.134077	13.0	
1	-1.002867	0.0	-0.531850	4.0	
2	0.451735	1.0	0.661316	11.0	
3	-1.230292	1.0	-0.981416	13.0	
4	0.455891	0.0	1.032540	4.0	
5	-0.656533	0.0	-0.036424	6.0	
6	1.548458	0.0	0.330115	0.0	
7	-0.558405	1.0	-0.973726	0.0	
8	-1.463490	0.0	1.125541	13.0	
9	-1.537375	1.0	-1.170496	0.0	
10	0.913513	1.0	0.739255	6.0	
11	-1.397687	1.0	0.783010	6.0	
12	0.516383	1.0	2.056581	8.0	
13	-1.814442	1.0	0.527900	1.0	
14	0.948146	1.0	-0.867888	7.0	

1	Item_Ider	Item_Weight	Item_Fat_	Item_Visi	Item_Type
2	FDW58	20.75	Low Fat	0.007565	Snack Foo
3	FDW14	8.3	reg	0.038428	Dairy
4	NCN55	14.6	Low Fat	0.099575	Others
5	FDQ58	7.315	Low Fat	0.015388	Snack Foo
6	FDY38		Regular	0.118599	Dairy
7	FDH56	9.8	Regular	0.063817	Fruits and
8	FDL48	19.35	Regular	0.082602	Baking Go
9	FDC48		Low Fat	0.015782	Baking Go
10	FDN33	6.305	Regular	0.123365	Snack Foo
11	FDA36	5.985	Low Fat	0.005698	Baking Go
12	FDT44	16.6	Low Fat	0.103569	Fruits and
13	FDQ56	6.59	Low Fat	0.105811	Fruits and
14	NCC54		Low Fat	0.171079	Health an
15	FDU11	4.785	Low Fat	0.092738	Breads
16	DRL59	16.75	LF	0.021206	Hard Drink
17	FDM24	6.135	Regular	0.079451	Baking Go

13.0 = Snack Food, 4.0 = Diary, 1.0 = bread and meat=10.0

For example we are taking 4 items type.

1. Snack foods
2. Dairy
3. Meat
4. Breads



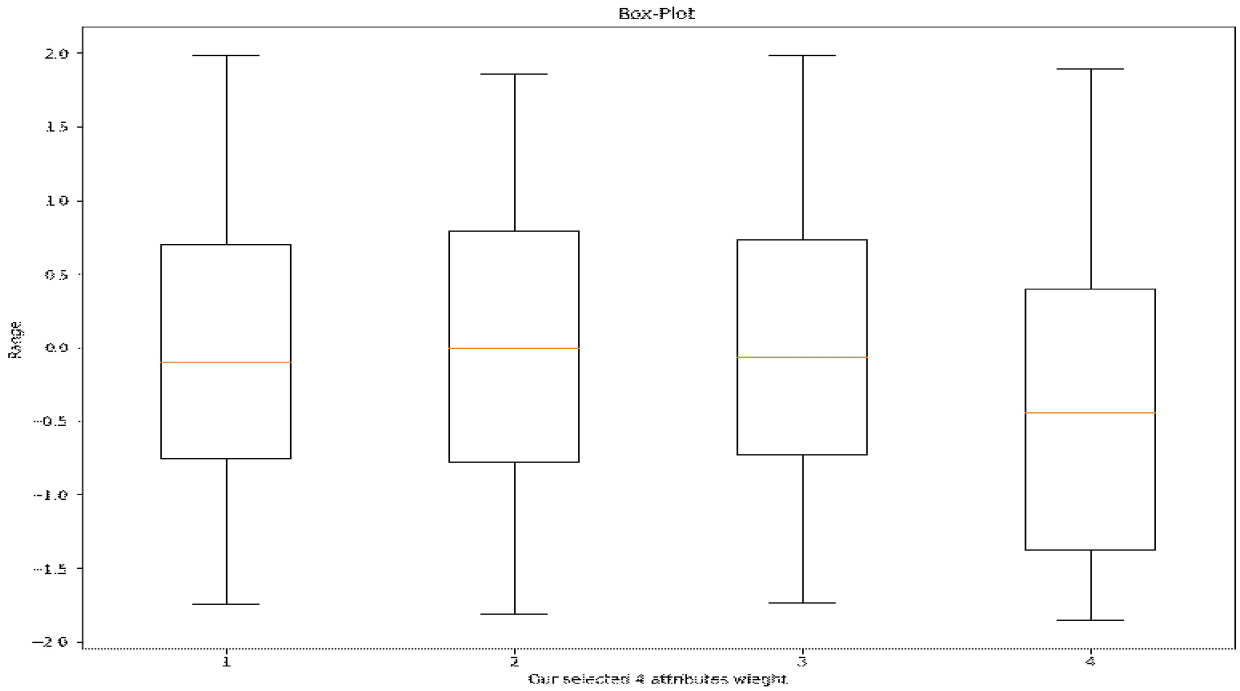
Analysis of box-plot

1. Snack foods:
    a. Average weight of this item is around 12.5 or a bit greater.
    b. Its max average value is around 15.5 (approximately)
    c. Its min average value is around 9.7 (approximately)
    d. its symmetric line shows that there is no skewness, which means no there are approximately equal number of values greater than mean and less than mean.
    e. There is no out liar value.
2. Dairy:
    a. Average weight of this item is around 12.
    b. Its max average value is around 16.25 (approximately)
    c. Its min average value is around 10 (approximately)
    d. its symmetric line shows that there is right skewned because it is toward lower half, the **number of items is greater** whose value is less than than mean.
    e. There is no out liar value.
3. Meat:
    a. Average weight of this item is around 12.5 or a bit greater.

b. Its max average value is around 15.5 (approximately)
c. Its min average value is around 9.8 (approximately)
d. its symmetric line shows that there is no skewness, which means no there are approximately equal number of values greater than mean and less than mean.
e. There is no out liar value.

4. Breads:
   a. Average weight of this item is around 11.25 or a bit greater.
   b. Its max average value is around 13.25 (approximately)
   c. Its min average value is around 6.5 (approximately)
   d. its symmetric line shows that there is left skewned because it is toward upper half, the **number of items is greater** whose value is greater than mean.
   e. There is no out liar value.

Now we are using standardized values.



Summary of using standardized and non-standardized values

There is no different in results except the range. code : https://github.com/ZeeWING-Projects/DM-Project/blob/main/Graphs-code/Box%20ploter.ipynb

Scatter plot

What is a Scatter Plot?

A scatter plot is a type of data visualization that shows the relationship between different variables. This data is shown by placing various data points between an x- and y-axis. Essentially, each of these data points looks “scattered” around the graph, giving this type of data visualization its name. Scatter plots can also be known as scatter diagrams or x-y graphs, and the point of using one of these is to determine if there are patterns or correlations between two variables.

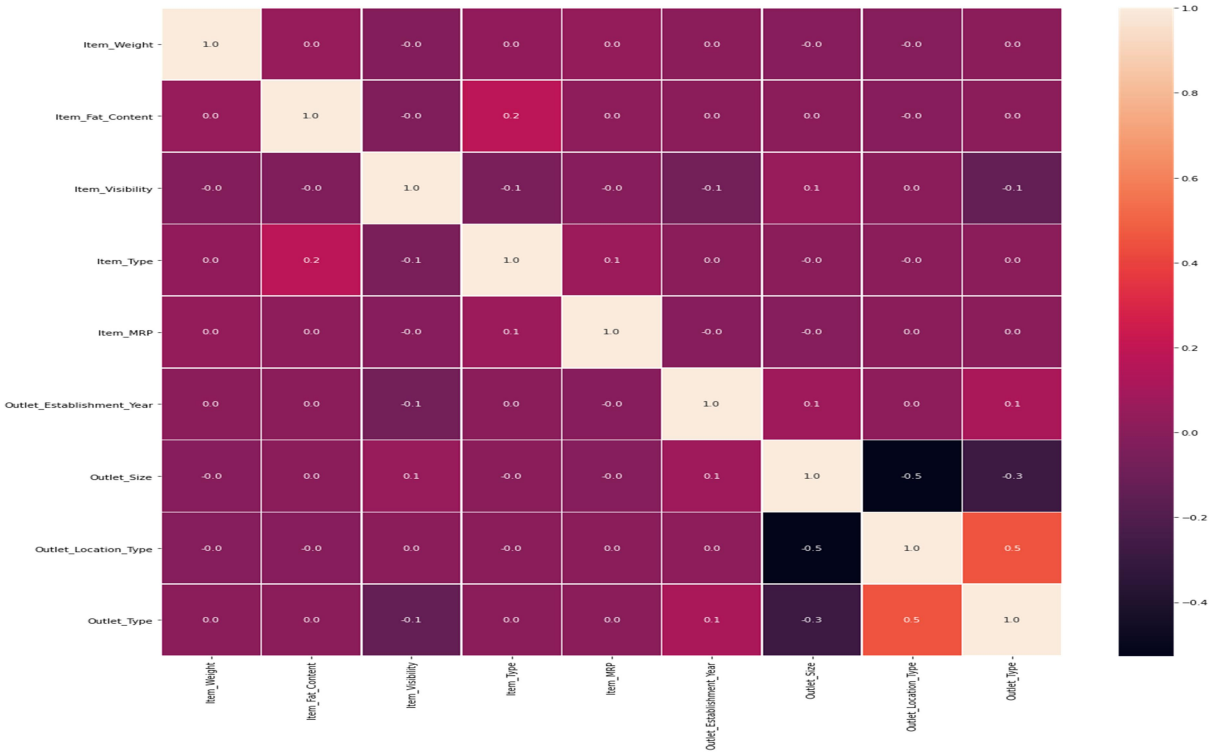
The patterns or correlations found within a scatter plot will have a few different features.

- Linear or Nonlinear: A linear correlation forms a straight line in its data points while a nonlinear correlation might have a curve or other form within the data points.
- Strong or Weak: A strong correlation will have data points close together while a weak correlation will have data points that are further apart.
- Positive or Negative: A positive correlation will point up (i.e., the x- and y-values are both increasing) while a negative correlation will point down (i.e., the x-values are increasing while the corresponding y-values are decreasing).

Source: https://visme.co/blog/scatter-plot/

Now we will try to find the co-relation with using scatter plot.

For the reference we have the already found co-relation, lets use that graph again.

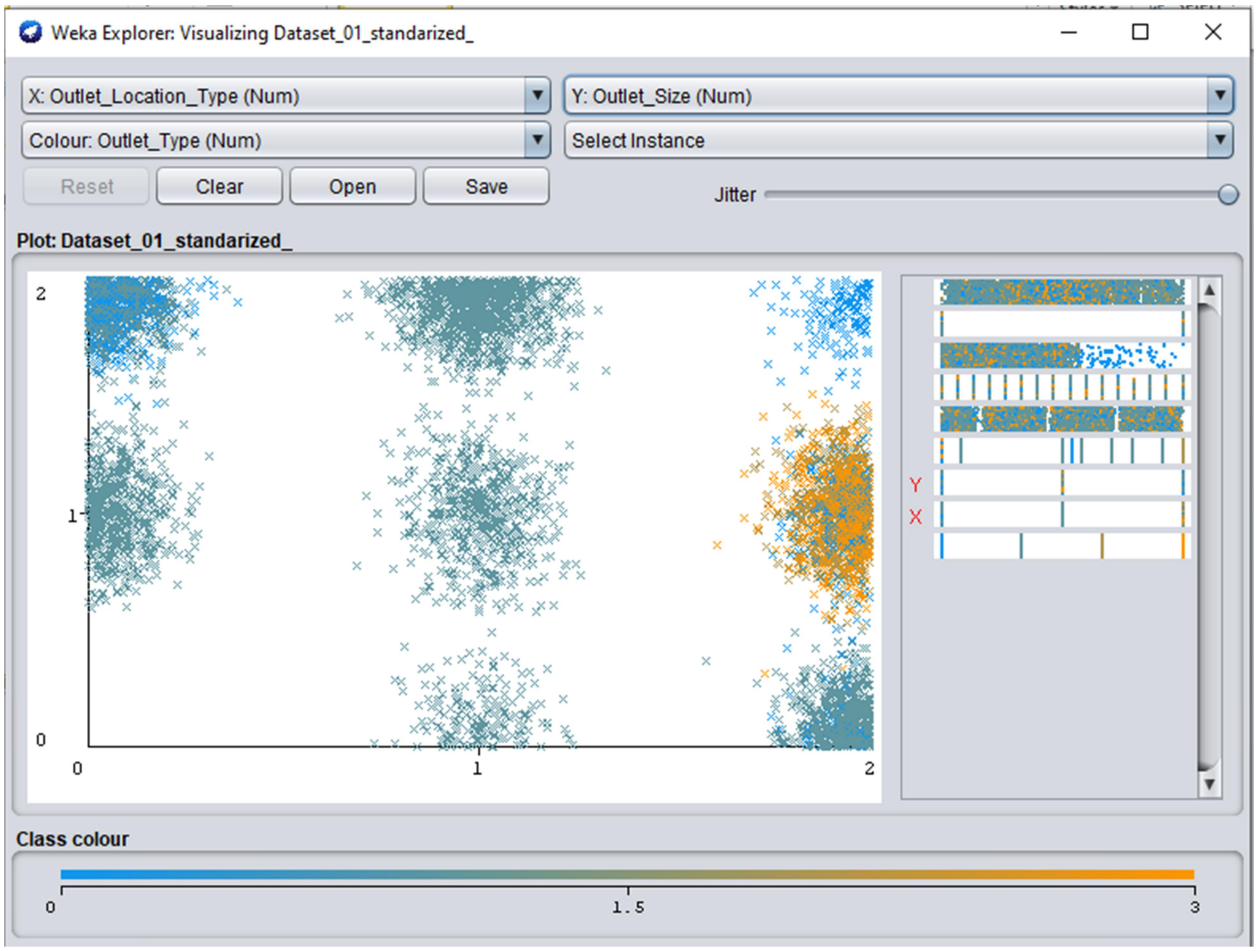


From this above graph we have found only few co-related relations.

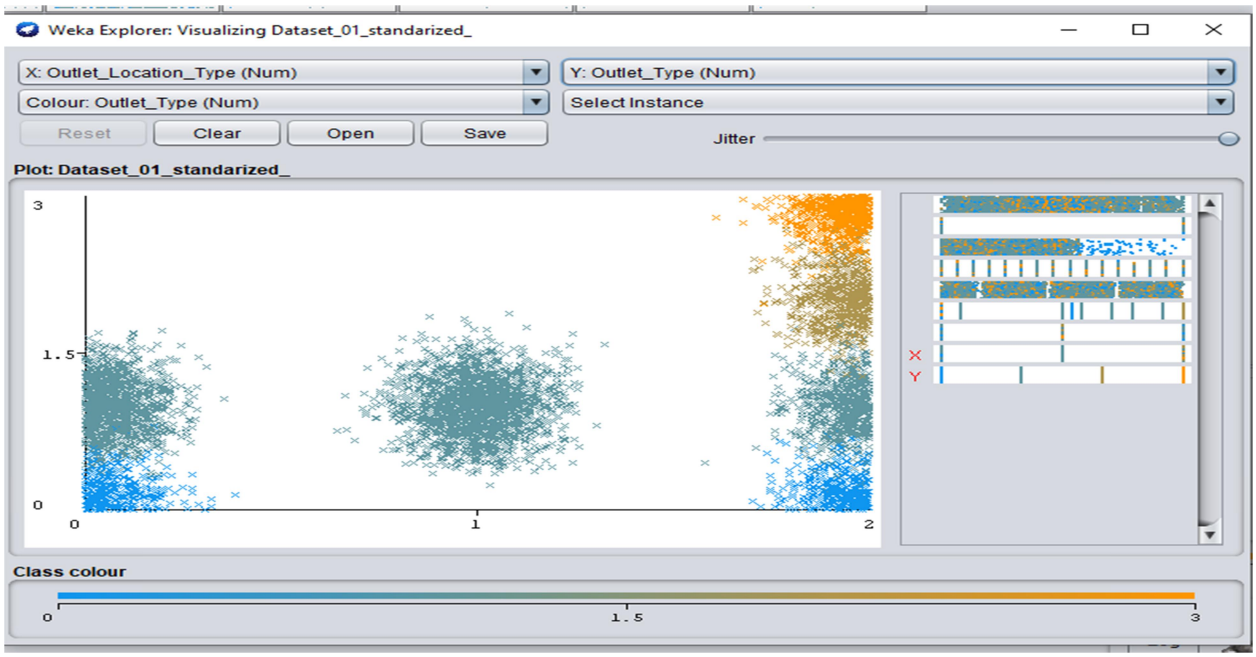
1. Negetive moderate co-relation between Outlet_size and outlet_location_type.
2. Positive moderate co-relation between outlet_location_type and outlet_type.
3. Very weak positive co-relation between item_fat_content and item_type.

Now lets see their scattered plots.

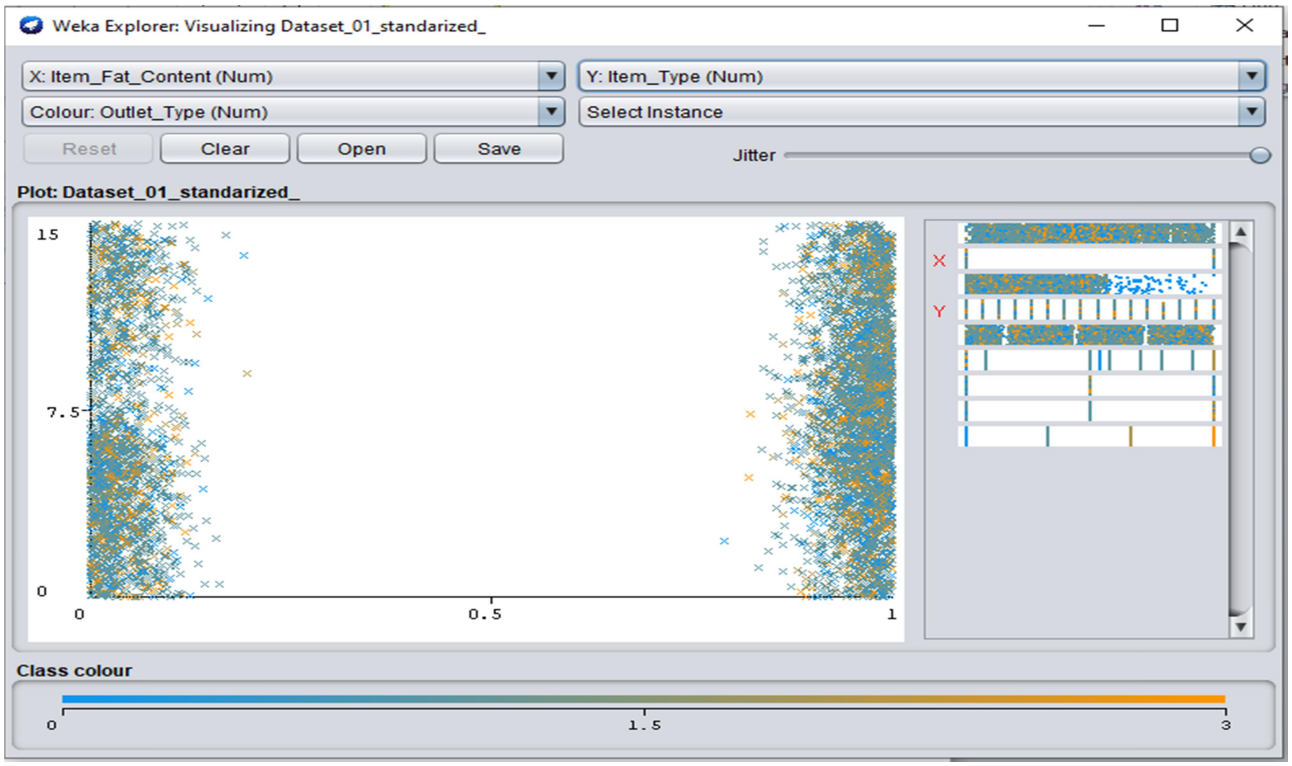
Negetive moderate co-relation between Outlet_size and outlet_location_type.



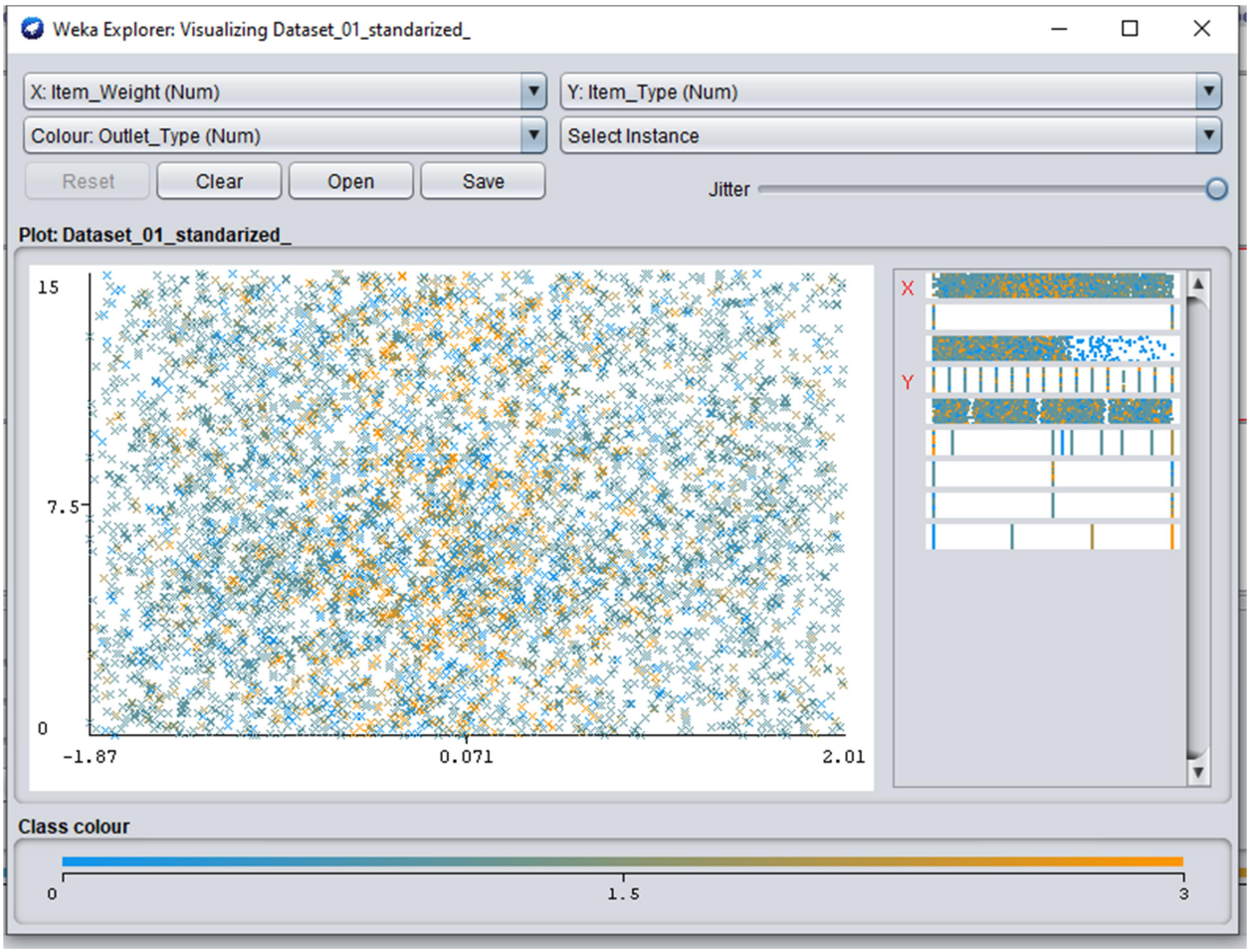
Positive very weak co-relation between outlet_location_type and outlet_type.



Exteremly weak positive co-relation between item_fat_content and item_type

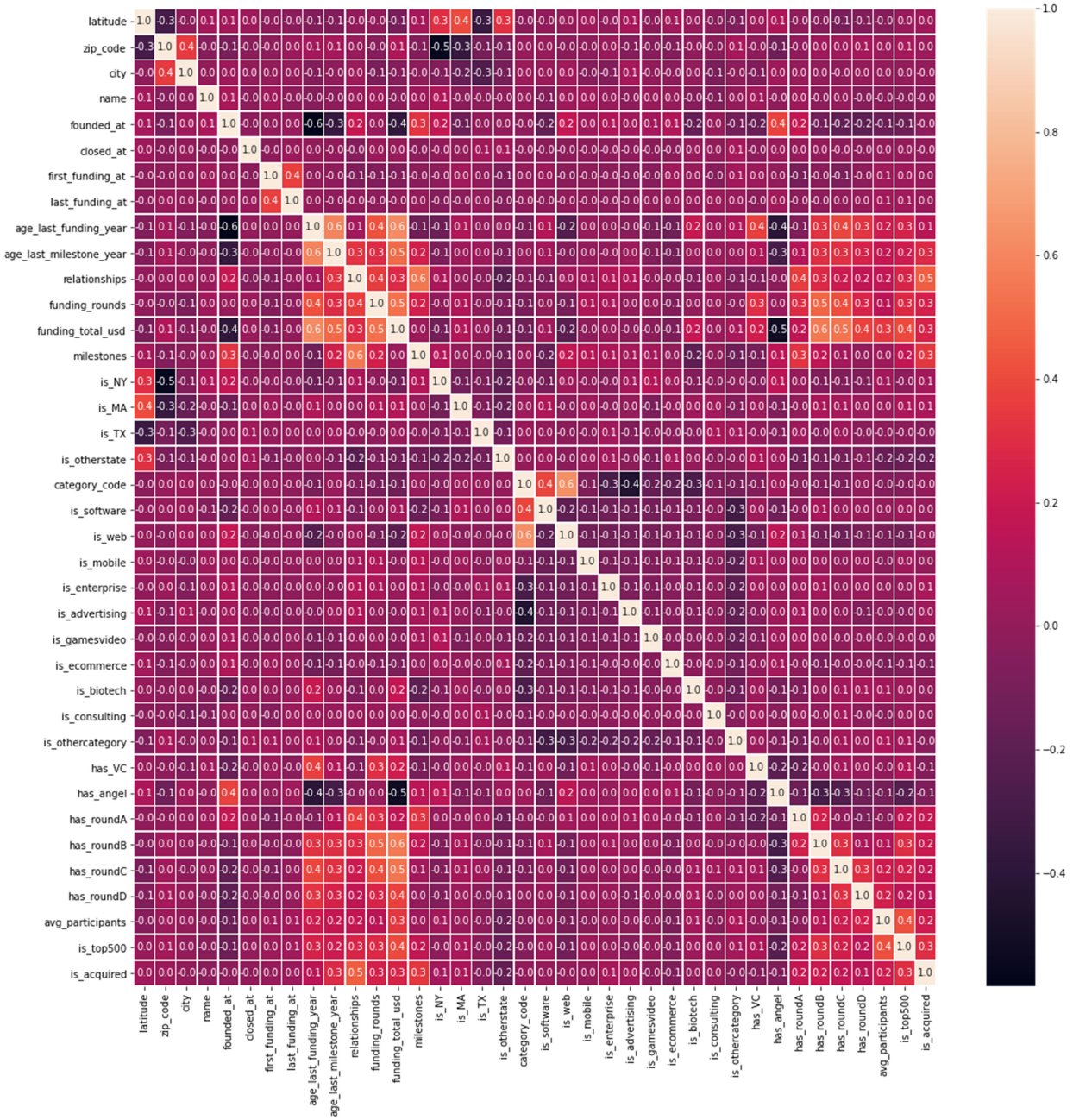


No relation

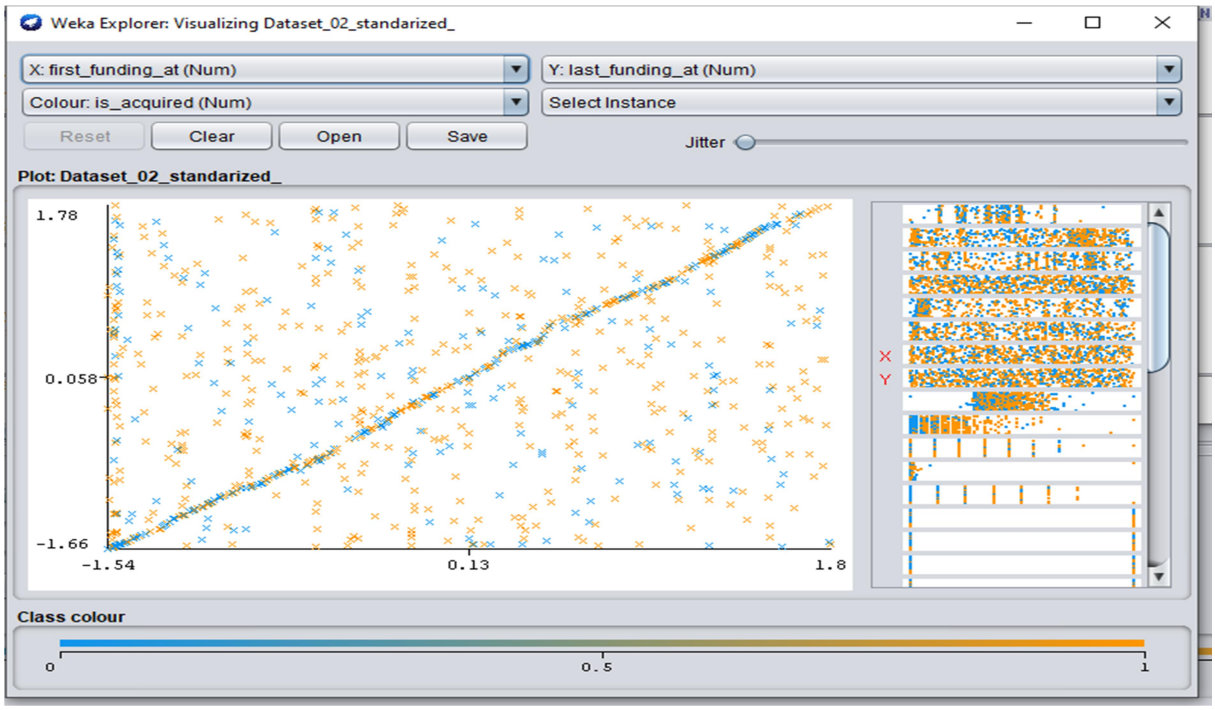

Weka Explorer: Visualizing Dataset_01_standarized_
X: Item_Weight (Num)
Y: Item_Type (Num)
Colour: Outlet_Type (Num)
Select Instance
Reset
Clear
Open
Save
Jitter
Plot: Dataset_01_standarized_
15
7.5
0
-1.87
0.071
2.01
X
Y
Class colour
0
1.5
3

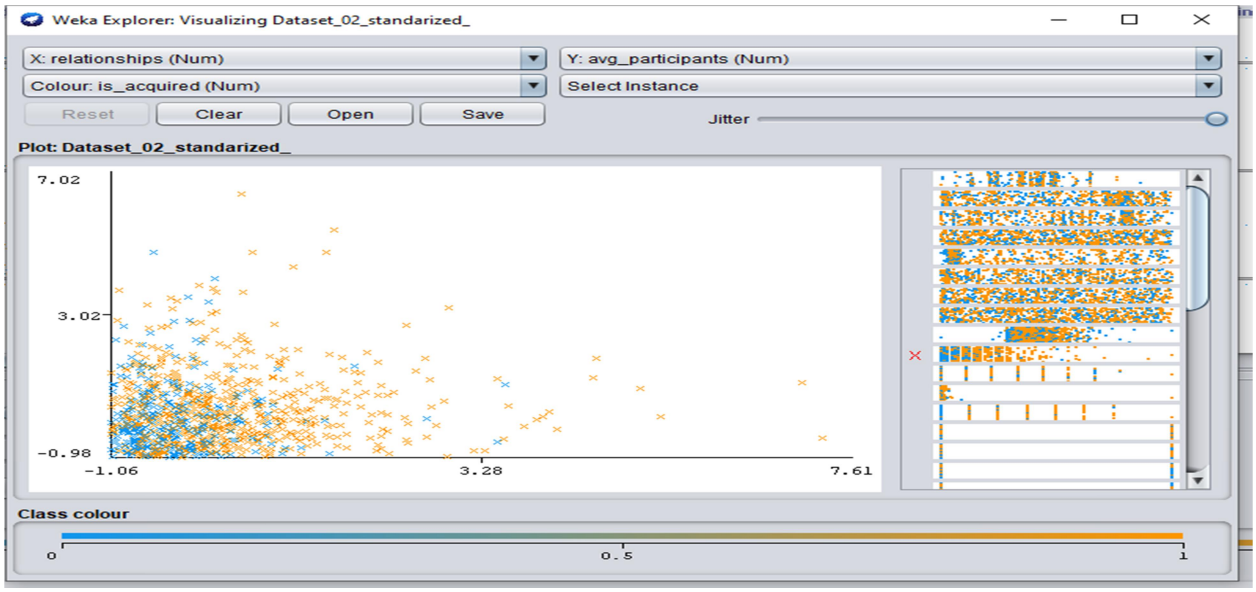
Now using the dataset two.

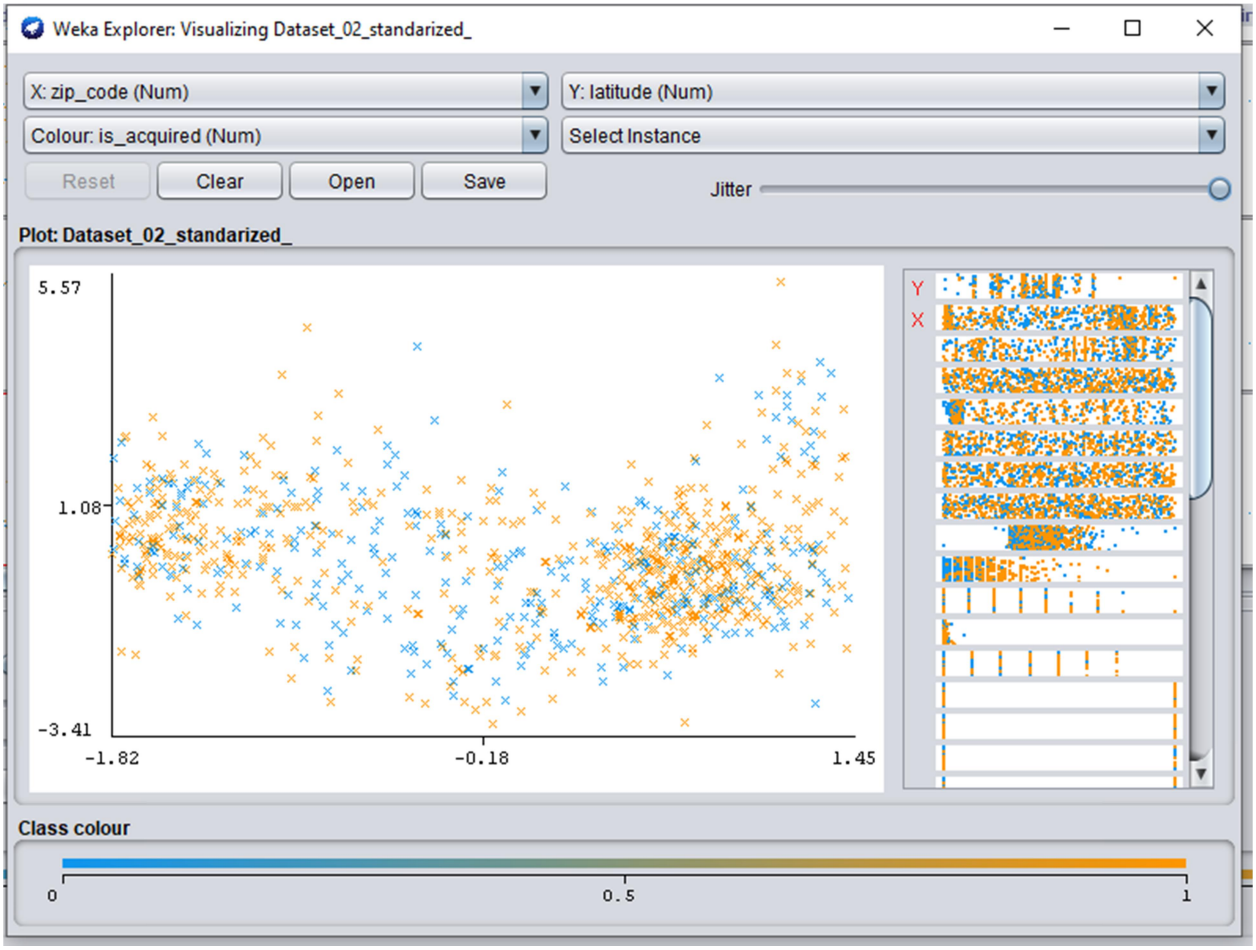
So just for reference lets see tha already computed graph.



We have following co-relations found in this graph.






Weka Explorer: Visualizing Dataset_02_standarized_
X: zip_code (Num)
Y: latitude (Num)
Colour: is_acquired (Num)
Select Instance
Reset
Clear
Open
Save
Jitter
Plot: Dataset_02_standarized_
5.57
1.08
-3.41
-1.82
-0.18
1.45
Y
X
Class colour
0
0.5
1

Applying classifier algorithms

Find the code at : https://github.com/ZeeWING-Projects/DM-Project/blob/main/Classification%20code/Classifcation-Algorithms.ipynb

Now we apply multiple type of classification algorithms on our pre-processed datasets. As we are using dataset-2 and its standardized version. While performing these classification algorithms we will note few things like we will notice the results for with varying different parameters like spilit size, providing non-normalized values and providing un smoted data (For just information note that we use somting when we have unequal number of records for each class). From word somted I mean smoting process is applied.

Following is the list of algorithms which we will be using in each test.

1. SVC
2. NuSVC
3. LinearSVC
4. KNeighborsClassifier
5. GaussianNB
6. RandomForestClassifier
7. ExtraTreesClassifier
8. DecisionTreeClassifier

Note that we have selected all columns which we got after pre-processing.

Test 01: [Different test-sizes, dataset Is normalized, and dataset is smoted]

Results: Following table shows the accuracy of all mentioned algorithms on different test size.

	Training Part					
	95%	90%	80%	60%	50%	30%
SGD Classifier	58.33%	52.50%	54.81%	52.30%	52.26%	48.21%
SVG	53.33%	59.17%	53.97%	52.30%	52.26%	47.97%
NuSVC	53.33%	59.17%	53.97%	52.30%	52.26%	47.97%
LinearSVC	46.67%	55.83%	61.51%	60.67%	47.74%	48.21%
KNeighborsClassifier	61.67%	64.17%	66.53%	65.90%	62.98%	61.00%
GaussianNB	41.67%	46.67%	44.77%	48.33%	49.75%	48.92%
Random Forest	81.67%	85.00%	81.17%	81.17%	81.24%	77.27%
Extra Trees	83.33%	84.17% %	80.33%	79.71%	80.57%	77.99%
Dedicion Tree	73.33%	78.33%	73.22%	71.97%	73.20%	67.82%

Summary of effect of change in test-size:

As from above table it is very clear to notice the effect of changing test size. As when the we spilit dataset in a way that training part is lower than testing part then it have lower accuracy. On other hand as table shows as we are increasing the training part then the accuracy is increasing for few algorithems and few algorithms are reducing their accuracy as training size is increasing and few are not affected that much. Its is also observed that few algorithms has reduced accuracy after 90%-80% training part.

Following are the best points of spilit where the specific algorithm gives best accuracy.

Best points of spiliting dataset for training and testing

Best point to spilit dataset			
	Test Part	Traning Part	Accuracy
SGD Classifier	5%	95%	54.81%
SVG	10%	90%	59.17%
NuSVC	10%	90%	59.17%
LinearSVC	20%	80%	61.51%
KNeighborsClassifier	20%	80%	66.53%
GaussianNB	50%	50%	49.75%
Random Forest	10%	90%	85.00%
Extra Trees	10%	90%	84.17%
Dedicion Tree	10%	90%	78.33%

These are the points of split which I will be need when I will choose any of above algorithms for classification on my dataset.

Wrost points of spiliting dataset for training and testing

Wrost point to spilit dataset			
	Test Part	Traning Part	Accuracy
SGD Classifier	70%	30%	58.33%
SVG	70%	30%	47.97%
NuSVC	70%	30%	47.97%
LinearSVC	5%	95%	46.67%
KNeighborsClassifier	70%	30%	61.00%
GaussianNB	5%	95%	41.67%
Random Forest	70%	30%	77.27%
Extra Trees	70%	30%	77.99%
Dedicion Tree	70%	30%	67.82%

These are the points of split which I will be have to never select when I will choose any of above algorithms for classification on my dataset.

Confusion matrix for best accuracy results of each algorithm

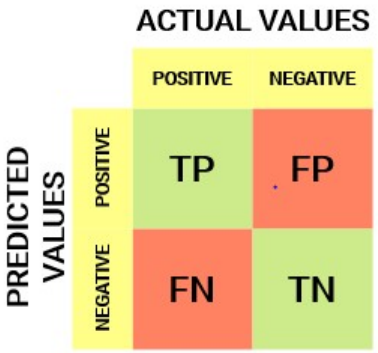
Before we move ahead for analyzing the confution matrix let's see few definitions which will help us in understanding the confusion matrix.

Source of information about confusion matrix :
https://www.analyticsvidhya.com/blog/2020/04/confusion-matrix-machine-learning/

What is a Confusion Matrix?

A Confusion matrix is an N x N matrix used for evaluating the performance of a classification model, where N is the number of target classes. The matrix compares the actual target values with those predicted by the machine learning model. This gives us a holistic view of how well our classification model is performing and what kinds of errors it is making.

For a binary classification problem, we would have a 2 x 2 matrix as shown below with 4 values:



Let's decipher the matrix:

- The target variable has two values: Positive or Negative
- The columns represent the actual values of the target variable
- The rows represent the predicted values of the target variable

But wait – what's TP, FP, FN and TN here? That's the crucial part of a confusion matrix. Let's understand each term below.

Understanding True Positive, True Negative, False Positive and False Negative in a Confusion Matrix

True Positive (TP)

- The predicted value matches the actual value
- The actual value was positive and the model predicted a positive value

True Negative (TN)

- The predicted value matches the actual value
- The actual value was negative and the model predicted a negative value

False Positive (FP) – Type 1 error

- The predicted value was falsely predicted
- The actual value was negative but the model predicted a positive value
- Also known as the Type 1 error

False Negative (FN) – Type 2 error

- The predicted value was falsely predicted
- The actual value was positive but the model predicted a negative value
- Also known as the Type 2 error

Let me give you an example to better understand this. Suppose we had a classification dataset with 1000 data points. We fit a classifier on it and get the below confusion matrix:

Confusion matrix example

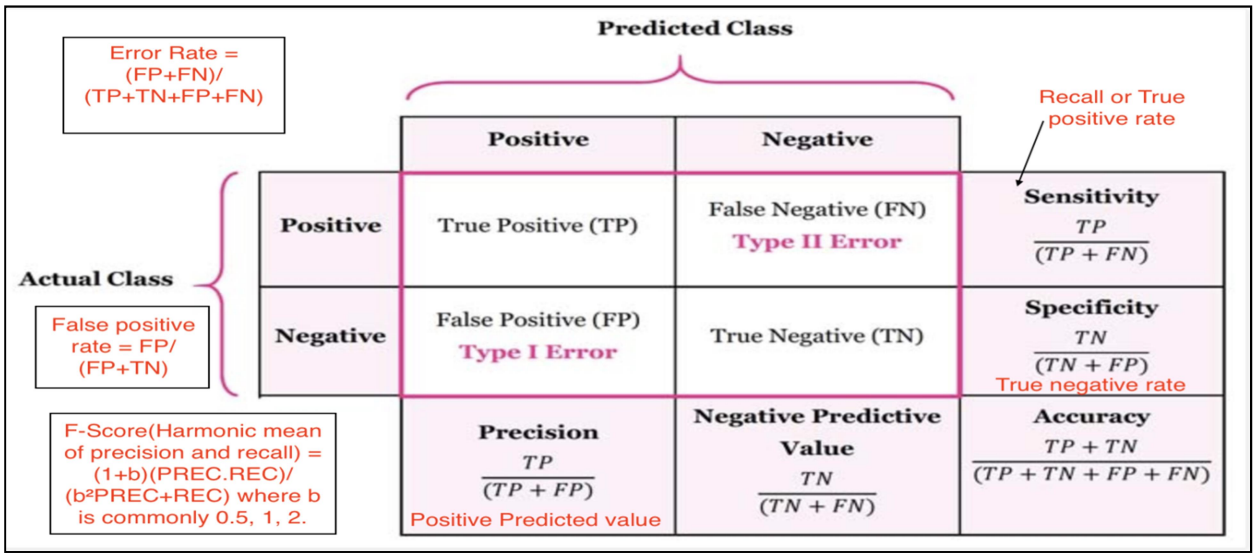
The different values of the Confusion matrix would be as follows:

- True Positive (TP) = 560; meaning 560 positive class data points were correctly classified by the model
- True Negative (TN) = 330; meaning 330 negative class data points were correctly classified by the model
- False Positive (FP) = 60; meaning 60 negative class data points were incorrectly classified as belonging to the positive class by the model
- False Negative (FN) = 50; meaning 50 positive class data points were incorrectly classified as belonging to the negative class by the model

This turned out to be a pretty decent classifier for our dataset considering the relatively larger number of true positive and true negative values.

Now we the above information on our generated confusion matrixes in best cases. in this section we will analyze the confusion matrix on best accuracy results.

So we will try to analyze following things



Sensitivity:

- True Positive recognition rate
  - **Sensitivity = TP/P**

Specificity

- True Negative recognition rate
  - **Specificity = TN/N**

Error rate

- *1 – accuracy*, or
  - **Error rate = (FP + FN)/All**

Precision

- exactness – what % of tuples that the classifier labeled as positive are actually positive
- $$precision = \frac{TP}{TP + FP}$$

Cross-validation (By using python)

MCC

- A

Actual \ Predicted	Control	Disease
Control	TN	FP
Disease	FN	TP

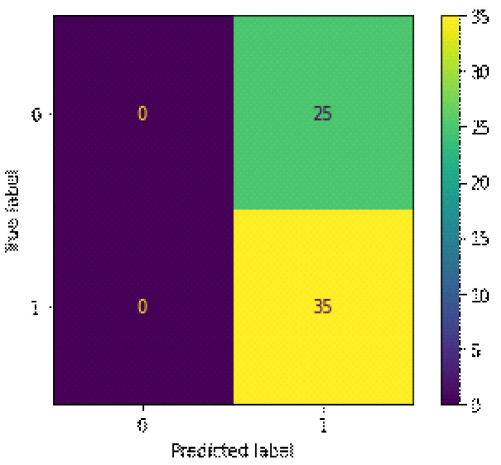
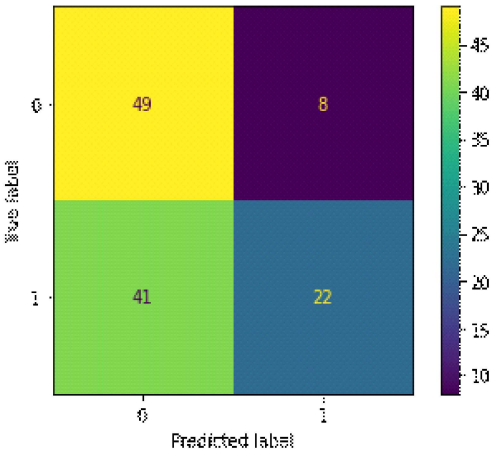
B

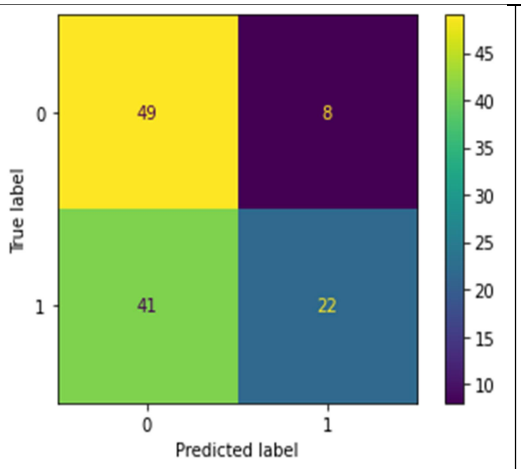
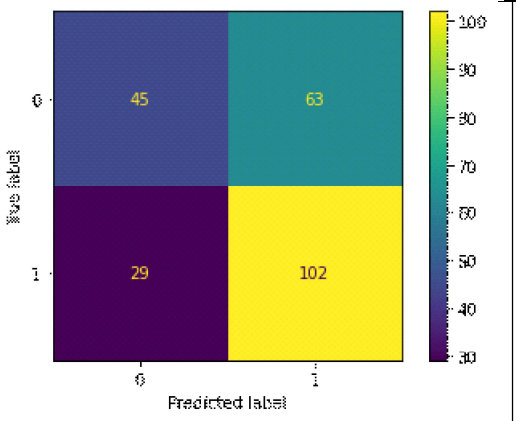
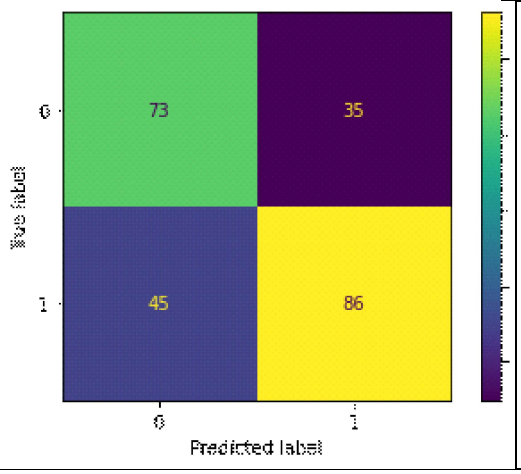
$$MCC = \frac{(TP \times TN) - (FP \times FN)}{\sqrt{(TP + FP)(TP + FN)(TN + FP)(TN + FN)}}$$

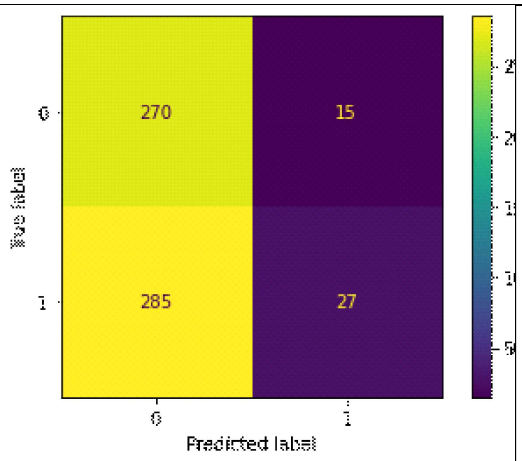
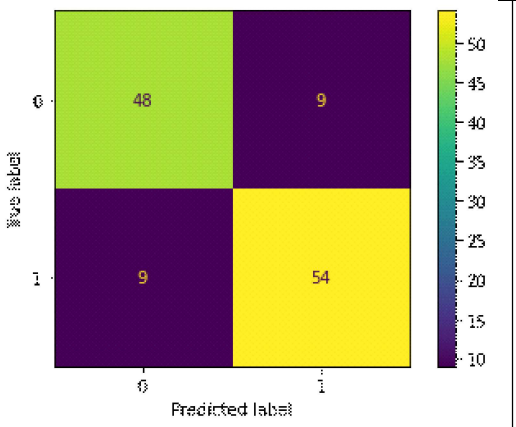
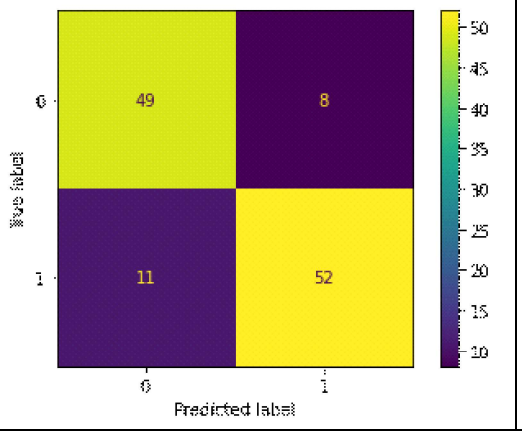
Note that I have used the dataset-2 for classification and following results are based on that dataset.

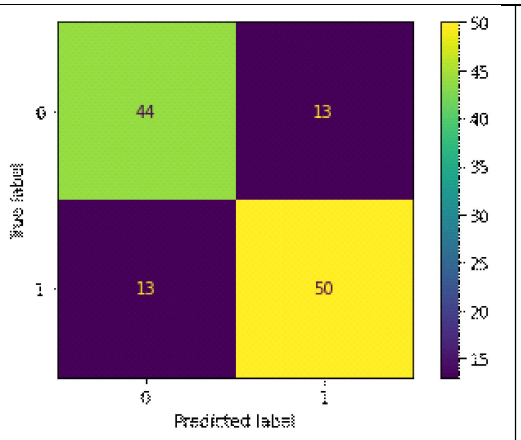
1 : “Acquired ” and 0: “Closed” Number of records while classification 1194

Classification results summary and details

Algorithem Name	Accuracy	Training part	Confusion matrix	Description Results
SGD Classifier	58.33 %	95%		Sensitivity is 100.0 % Specificity is 0.0 % Percision is 58.333 % MCC is 0.352 Error rate is 0.417 %
SVC	59.17 %	90%		Sensitivity is 34.921 % Specificity is 85.965 % Percision is 73.333 % -ve prid: are 54.444 % Calc: Accuracy is 59.167 % MCC is 0.241 Error rate is 0.408 % Cross validation score: 64.68% (+/- 0.39%)

NuSVC	59.17 %	90%		Sensitivity is 34.921 % Specificity is 85.965 % Percision is 73.333 % -ve prid: are 54.444 % Calc: Accuracy is 59.167 % MCC is 0.241 Error rate is 0.408 % Cross validation score: 57.74% (+/- 12.39%)
LinearSVC	61.51 % %	80%		Sensitivity is 77.863 % Specificity is 41.667 % Percision is 61.818 % -ve prid: are 60.811 % Calc Accuracy is 61.506 % MCC is 0.21 Error rate is 0.385 % Cross validation score: 42.13% (+/- 27.06%)
KNeighborsClassifier	66.53 %	80%		Sensitivity is 65.649 % Specificity is 67.593 % Percision is 71.074 % -ve prid: are 61.864 % Calc Accuracy is 66.527 % MCC is 0.331 Error rate is 0.335 % Cross validation score: 63.06% (+/- 5.59%)

Gaussian NB	49.75 %	50%		Sensitivity is 8.654 % Specificity is 94.737 % Percision is 64.286 % -ve prid: are 48.649 % Calc Accuracy is 49.749 % MCC is 0.066 Error rate is 0.503 % Cross validation score: 42.67% (+/- 21.14%)
Random ForestClassifier	85.00 %	90%		Sensitivity is 85.714 % Specificity is 84.211 % Percision is 85.714 % -ve prid: are 84.211 % Calc Accuracy is 85.0 % MCC is 0.699 Error rate is 0.15 % Cross validation score: 79.31% (+/- 2.50%)
ExtraTreesClassifier	84.17 % %	90%		Sensitivity is 82.54 % Specificity is 85.965 % Percision is 86.667 % -ve prid: are 81.667 % Calc Accuracy is 84.167 % MCC is 0.684 Error rate is 0.158 % Cross validation score: 75.84% (+/- 0.79%)

Decision TreeClassifier	78.33 %	90%		Sensitivity is 79.365 % Specificity is 77.193 % Percision is 79.365 % -ve prid: are 77.193 % Calc Accuracy is 78.333 % MCC is 0.566 Error rate is 0.217 % Cross validation score: 70.10% (+/- 5.29%)

Following is the piece of code which I have used for calculating the result.

Find code at : https://github.com/ZeeWING-Projects/DM-Project/blob/main/Confusion%20matrix%20calculator/Confusion%20matrix%20calculator.ipynb

```
import numpy as np
import math
```

```
#             Pridicted  Y     N
#       Actual      Y   [TP    FN]
#       Actual      N   [FP    TN]


TP = 50
FN = 13
FP = 13
TN = 44

P=(TP+FN)
Sensitivity = (TP/P)*100

P=(TN+FP)
Specificity=(TN/P)*100

P=(TP+FP)
Percision = (TP/P)*100
P=(TN+FN)
Negative_Prid_Value = (TN/P)*100

N = TP+TN
P=TP+TN+FP+FN

Accuracy = (N/P)*100
Error_rate = (1-(N/P))

N = ((TP*TN)-(FP*FN))
P = math.sqrt((TP+FP)*(TP+FN)*(TN+FP)*(TN+FN))

MCC = (N/P)

print("Sensitivity is {} % ".format(round(Sensitivity,3)))
print("Specificity is {} %".format(round(Specificity,3)))
print("Percision is {} %".format(round(Percision,3)))
print("-ve prid:  are {} %".format(round(Negative_Prid_Value,3)))
print("Calc Accuracy is {} %".format(round(Accuracy,3)))
print("MCC is {} ".format(round(MCC,3)))
print("Error rate is {} %".format(round(Error_rate,3)))
```

The End