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 algorithems.

**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://www.kaggle.com/brijbhushannanda1979/bigmart-sales-data>

**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> )

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

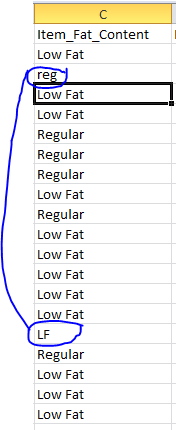
## Dataset 1 (big-mart-sales):

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

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

### 1.1 Matching field’s values:

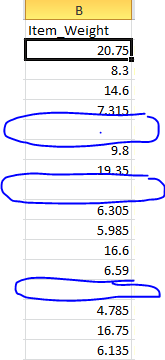
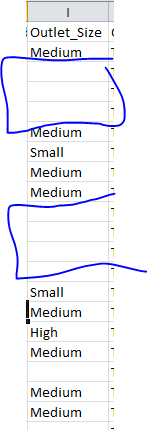
There are few fields in dataset which contain same data with different names, so we need to make the same. For example we have attribute item\_Fat\_Content which contain two labels, Low Fat and Regular, but for representing this same value “LF” and “reg” are used so we need to remove these shortcuts.



### 1.2Filling out missing values:

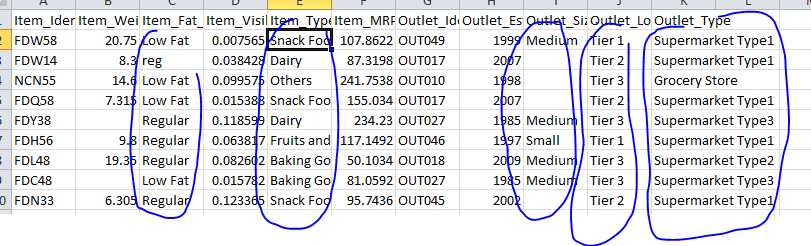
In our dataset we have a bunch of attributes having missing values. And we have to fill them by using well known pre-processing techniques. For example for numeric attribute we have methods like by using median, mean and mode and for ordinal attributes we will use some built in functions of python, like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

For example we have some attributes with missing values.

### 1.4 Conversion of string values to numeric values.

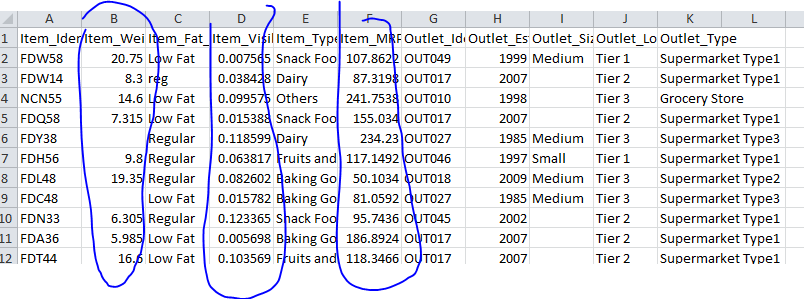
We have some values which are in string form so we need to transform them in numeric form. Since we have following values which need to be in numeric form.



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.



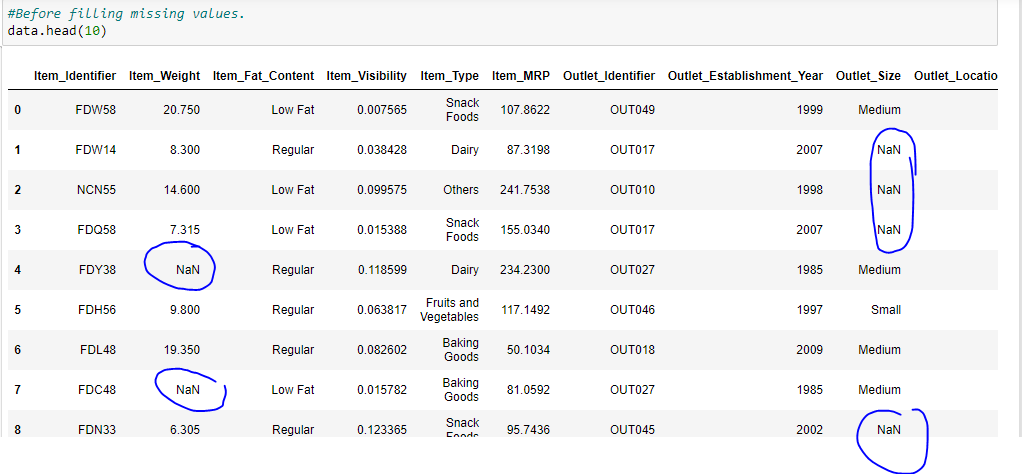
As highlighted values are those one which need to be standardized because this will affect the co-relation graph, which will be drawn later on.

### 1.5Implementation of above stated issues

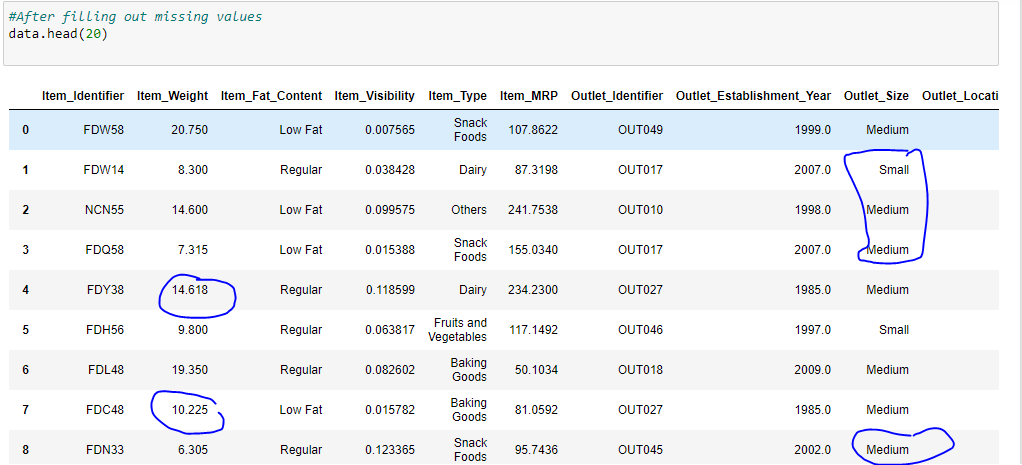
So we have a real dataset having some missing values and some other issues like **conversion of ordinal values to numeric form** so we need to perform above stated steps of pre-processing.

Find Code at <https://github.com/ZeeWING-Projects/DM-Project/blob/main/Preprocessing-Code/Dataset-1_pre-processing.py> )

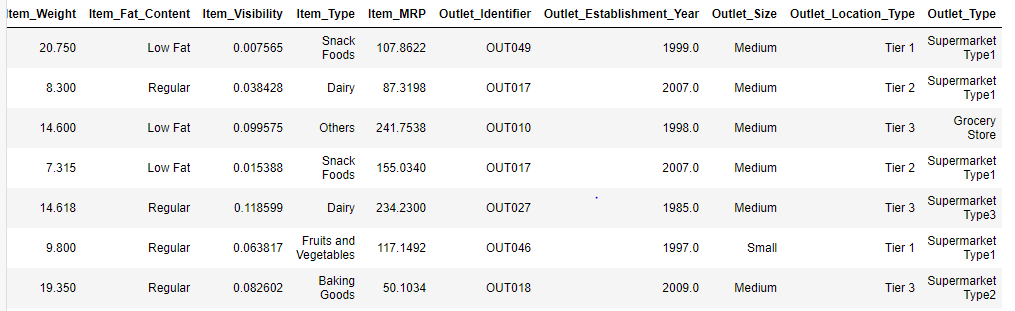
#### Before filling missing values



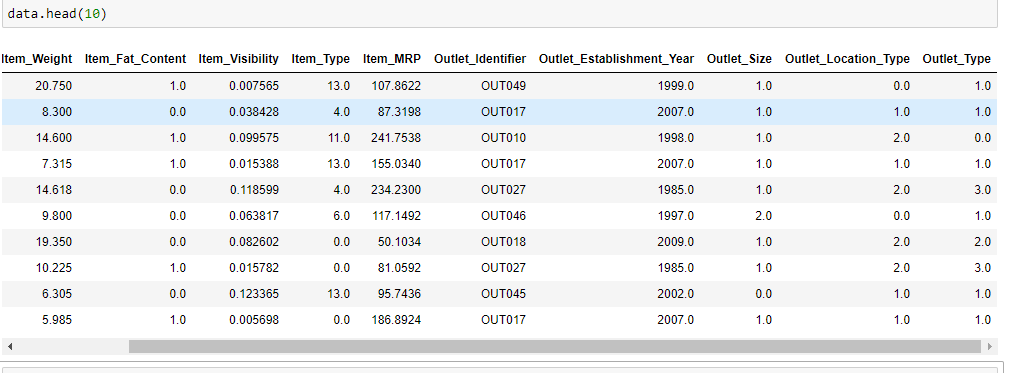
#### After filling missing values



#### Before converting string values to numeric form

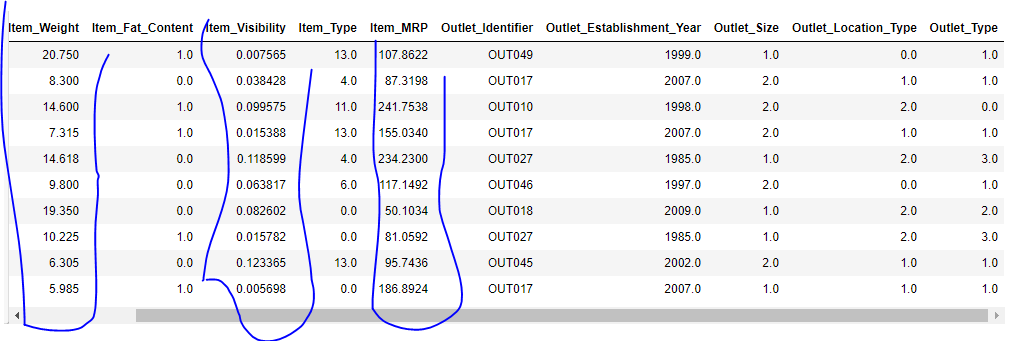


#### After converting string values to numeric form

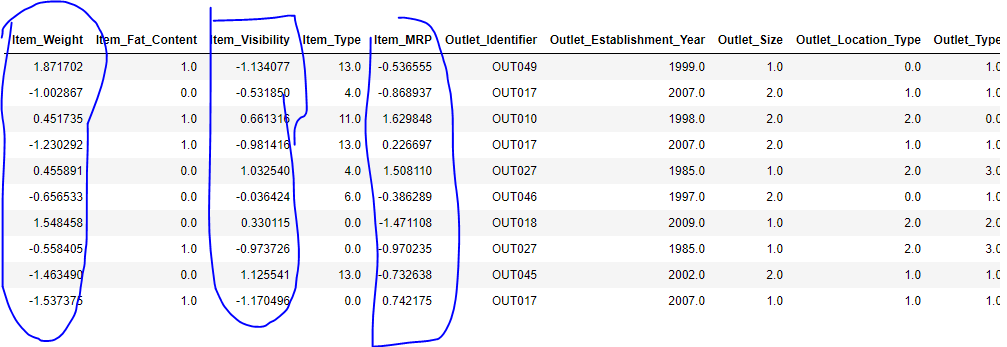


#### Before standardization of values

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> )

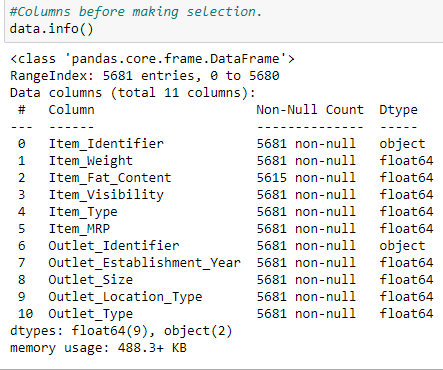


#### After standardization of values



#### Now making selection of features

#### Columns before selection



#### 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/Pre-procssing/Dataset-1%20Pre-processed/Dataset\_01\_standarized\_.csv**](https://github.com/ZeeWING-Projects/DM-Project/blob/Pre-procssing/Dataset-1%20Pre-processed/Dataset_01_standarized_.csv)

**Dataset (non-standardized values) :** [**https://github.com/ZeeWING-Projects/DM-Project/blob/Pre-procssing/Dataset-1%20Pre-processed/Dataset\_01\_non\_standarized\_.csv**](https://github.com/ZeeWING-Projects/DM-Project/blob/Pre-procssing/Dataset-1%20Pre-processed/Dataset_01_non_standarized_.csv)

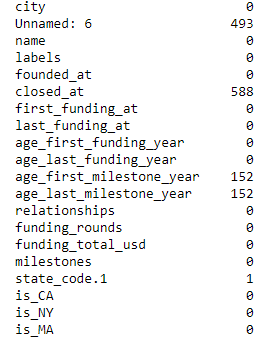
## 2. Applying some pre-processing steps (Data set 2):

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

### 2.2Filling out missing values:

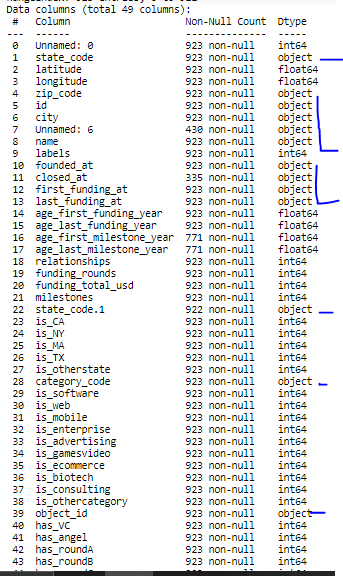
In our dataset we have a bunch of attributes having missing values. And we have to fill them by using well known pre-processing techniques. For example for numeric attribute we have methods like by using median, mean and mode and for ordinal attributes we will use some built in functions of python, like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



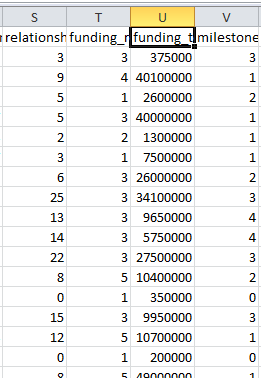
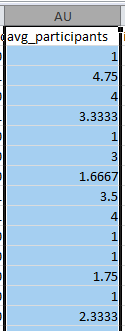
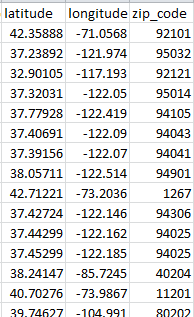
### 2.2 Conversion of string values to numeric values.

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.



### 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.

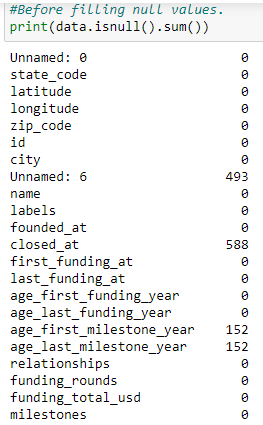
As highlighted values are those one which need to be standardized because this will affect the co-relation graph, which will be drawn later on. And we will have some other columns aswell which will be in numeric form after conversion.

### 1.5Implementation of above stated issues

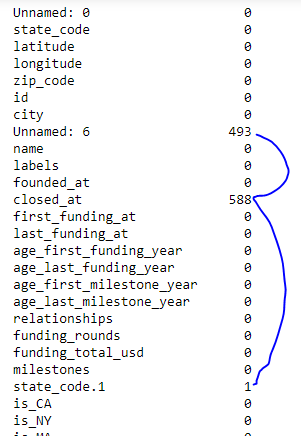
So we have a real dataset having some missing values and some other issues like **conversion of ordinal values to numeric form** so we need to perform above stated steps of pre-processing.

Find Code at :( <https://github.com/ZeeWING-Projects/DM-Project/blob/main/Preprocessing-Code/Dataset-2_pre-processing.py> )

#### Before filling missing values

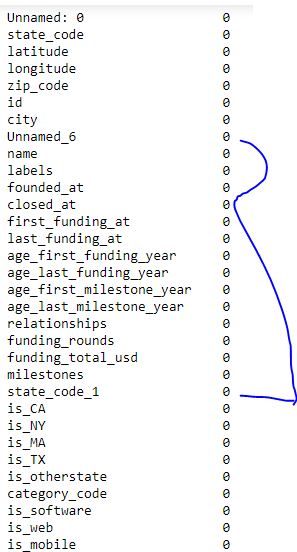


#### After filling missing values



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)



**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

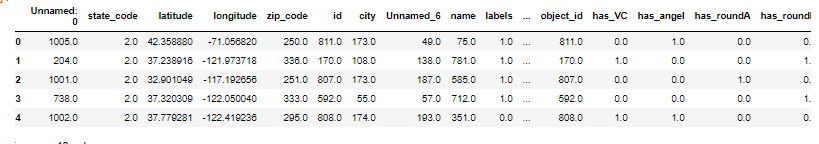
47 is\_top500 923 non-null float64

48 is\_acquired 923 non-null int64

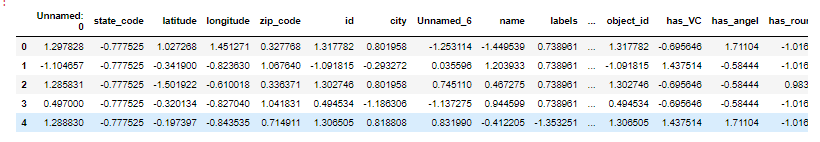
**Now we have successfully converted the string value to numeric form.**

#### 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> )



#### After standardization of values

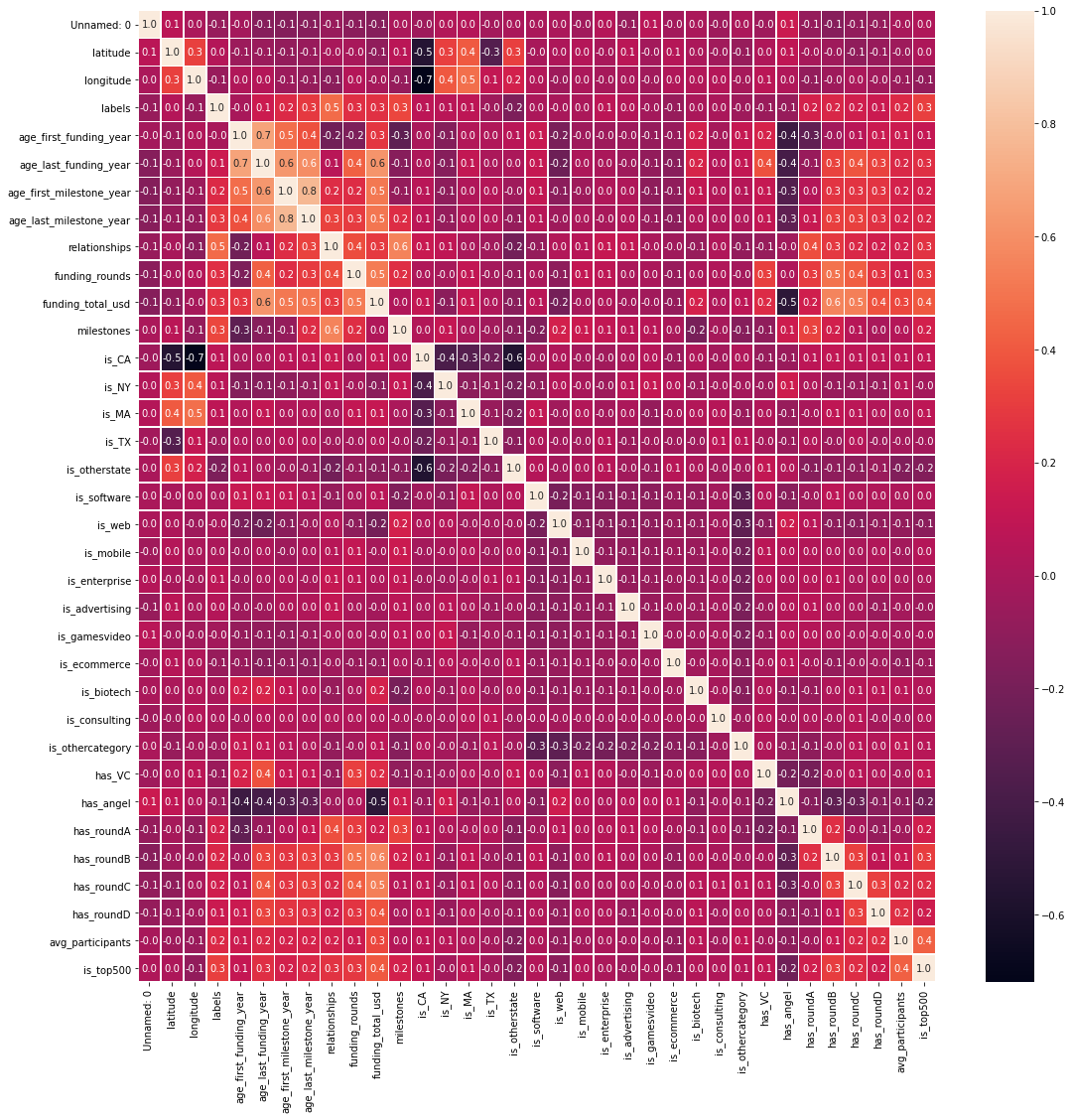


#### Now making selection of features

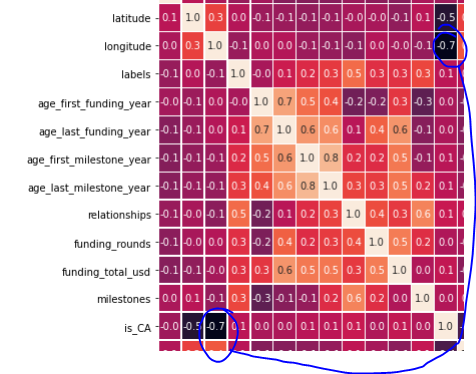
#### Columns before selection

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

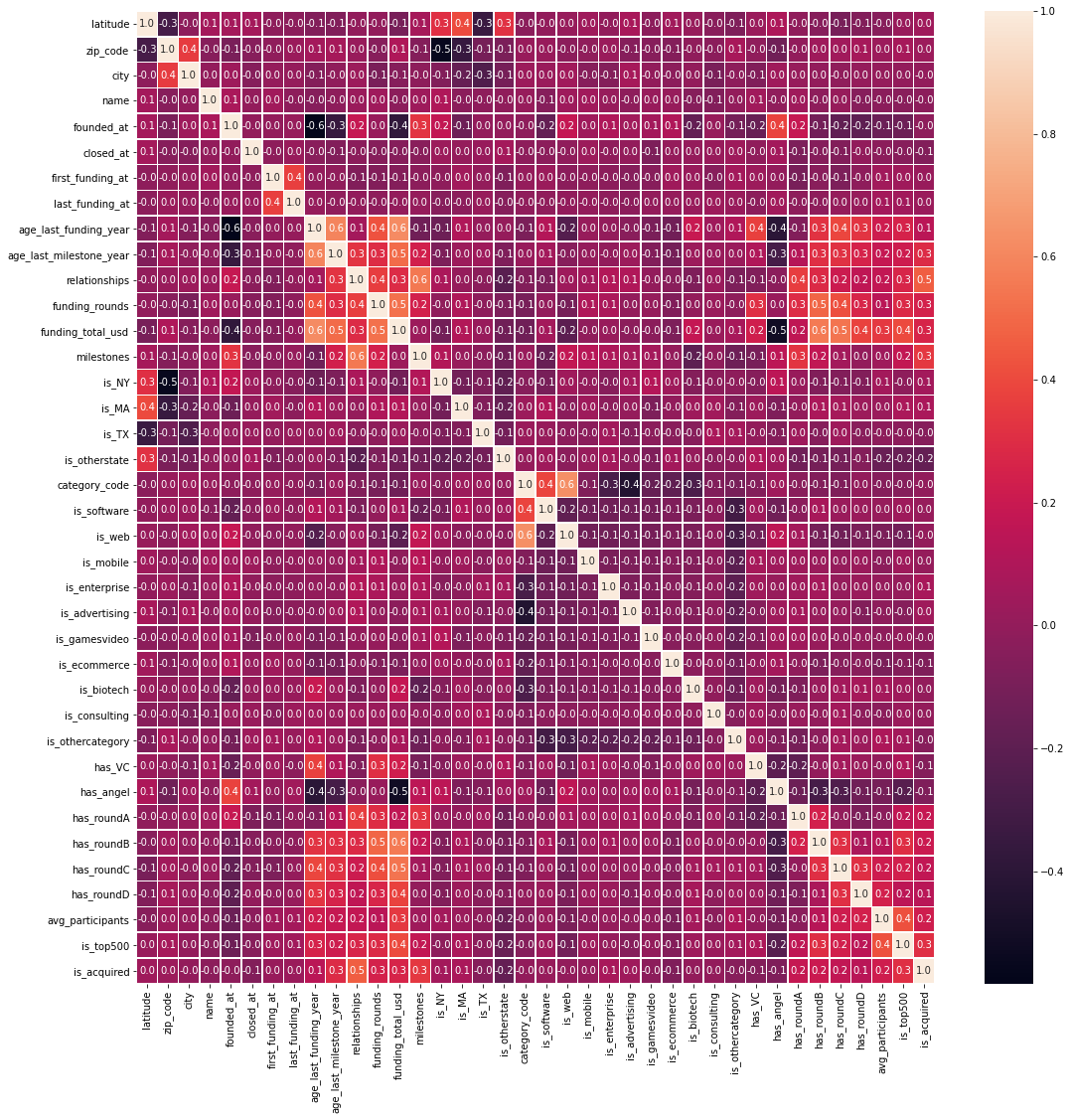
# • labels

# • 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.



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.

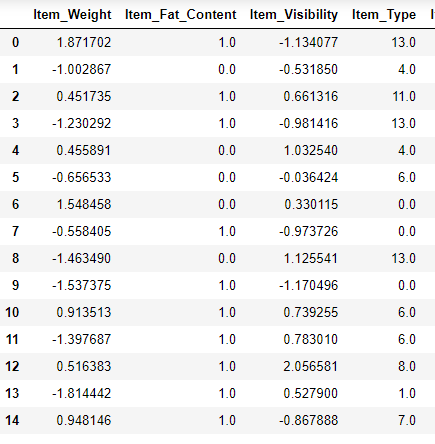
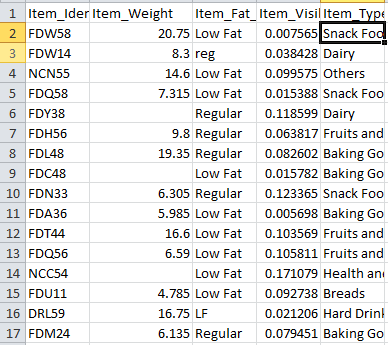
* Detect outlier values.
* Mean tendency of values.
* Symmetry of data.

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.

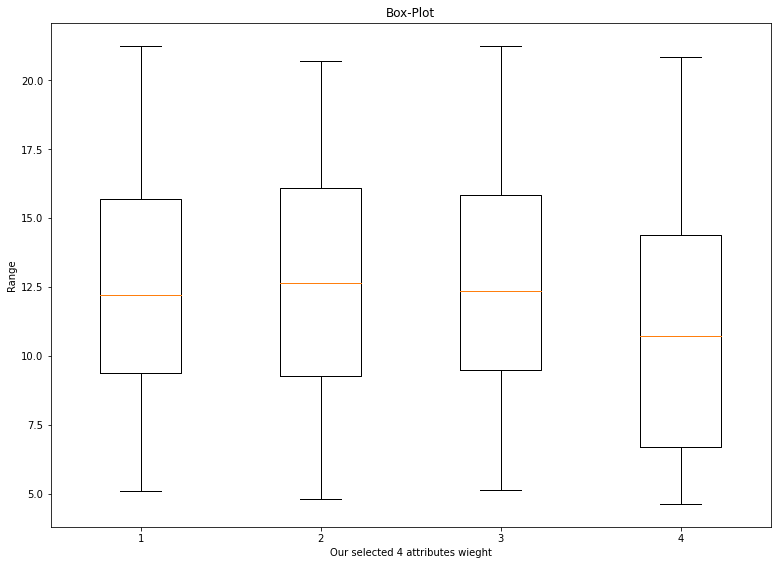
 

**13.0 = Snack Food, 4.0 = Diary, 1.0 = bread and meat=10.0**

For example we are taking 4 items type.

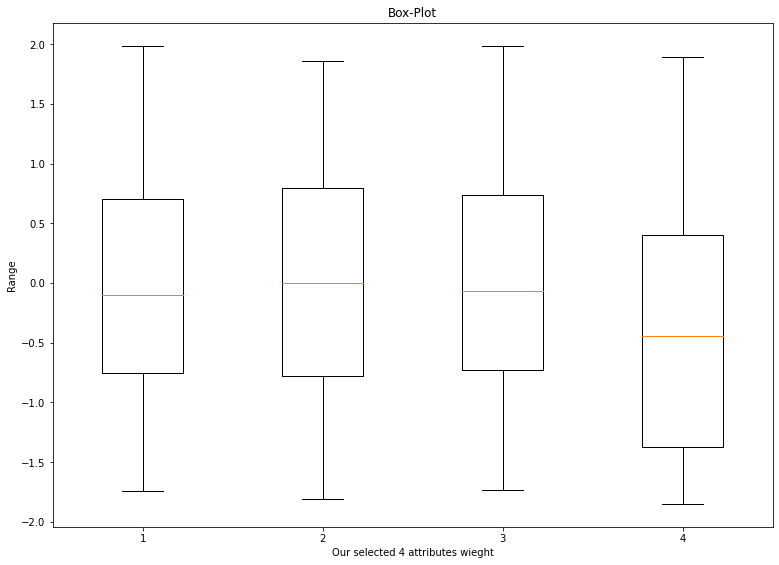
1. Snack foods
2. Dairy
3. Meat
4. Breads

Analysis of box-plot



1. Snack foods:
   1. Average weight of this item is around 12.5 or a bit greater.
   2. Its max average value is around 15.5 (approximately)
   3. Its min average value is around 9.7 (approximately)
   4. 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.
   5. There is no out liar value.
2. Dairy:
   1. Average weight of this item is around 12.
   2. Its max average value is around 16.25 (approximately)
   3. Its min average value is around 10 (approximately)
   4. 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.
   5. There is no out liar value.
3. Meat:
   1. Average weight of this item is around 12.5 or a bit greater.
   2. Its max average value is around 15.5 (approximately)
   3. Its min average value is around 9.8 (approximately)
   4. 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.
   5. There is no out liar value.
4. Breads:
   1. Average weight of this item is around 11.25 or a bit greater.
   2. Its max average value is around 13.25 (approximately)
   3. Its min average value is around 6.5 (approximately)
   4. 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.
   5. 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/Graph-ploting-box-plot.py>

# 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.

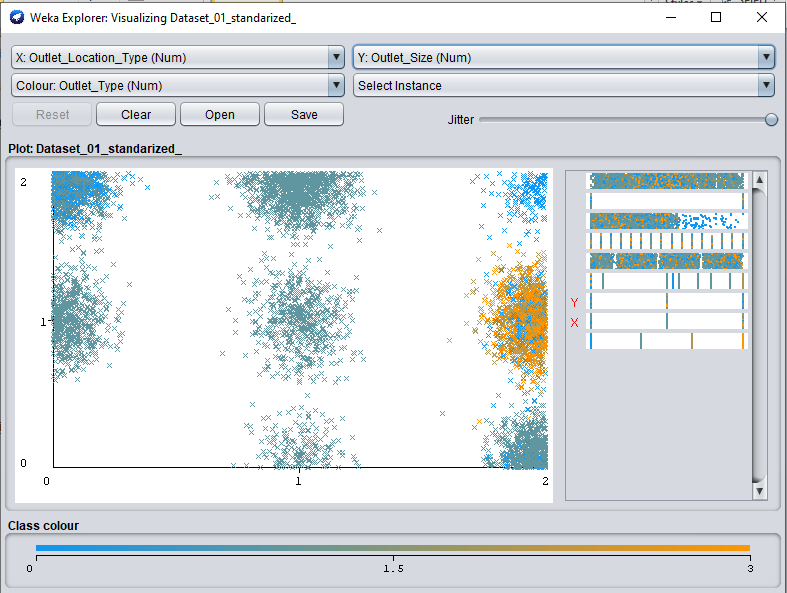


From this above graph we have found only few co-related relations.

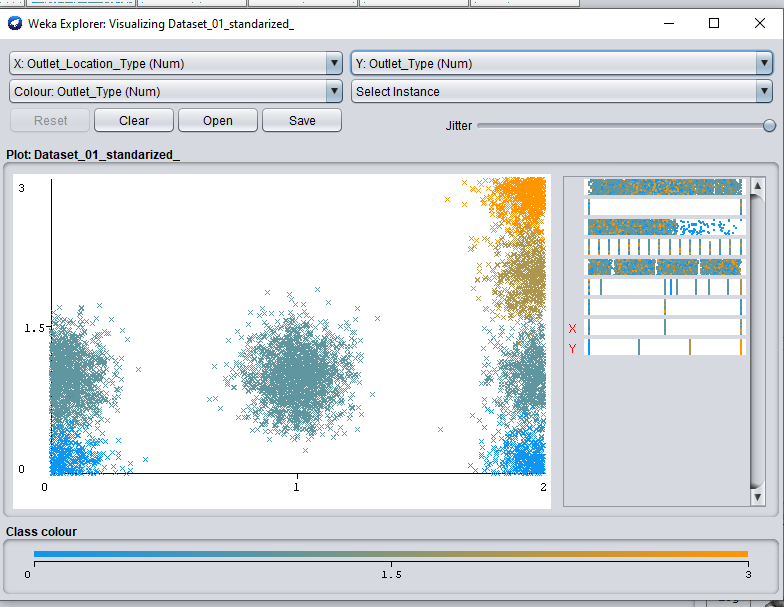
1. Negetive moderate co-relation between Outlet\_size and outlet\_location\_type.
2. Positive moderate co-relation between outlet\_location\_type and outlet\_type.
3. Very weak positive co-relation between item\_fat\_content and item\_type.

Now lets see their scattered plots.

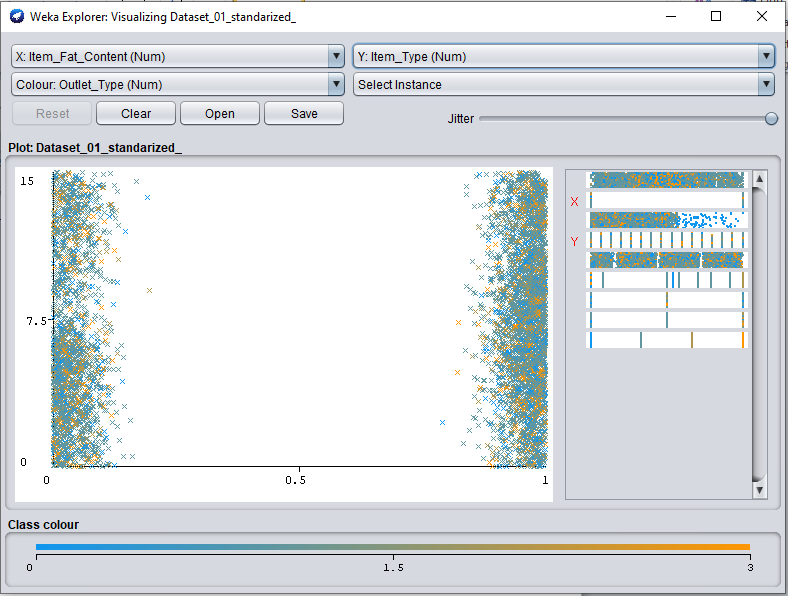
Negetive moderate co-relation between Outlet\_size and outlet\_location\_type.



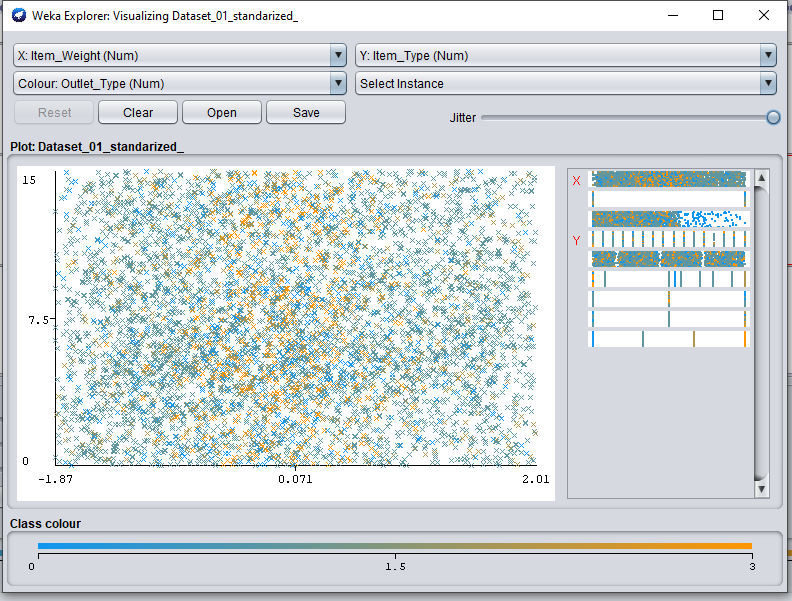
Positive very weak co-relation between outlet\_location\_type and outlet\_type.



Exteremly weak positive co-relation between item\_fat\_content and item\_type

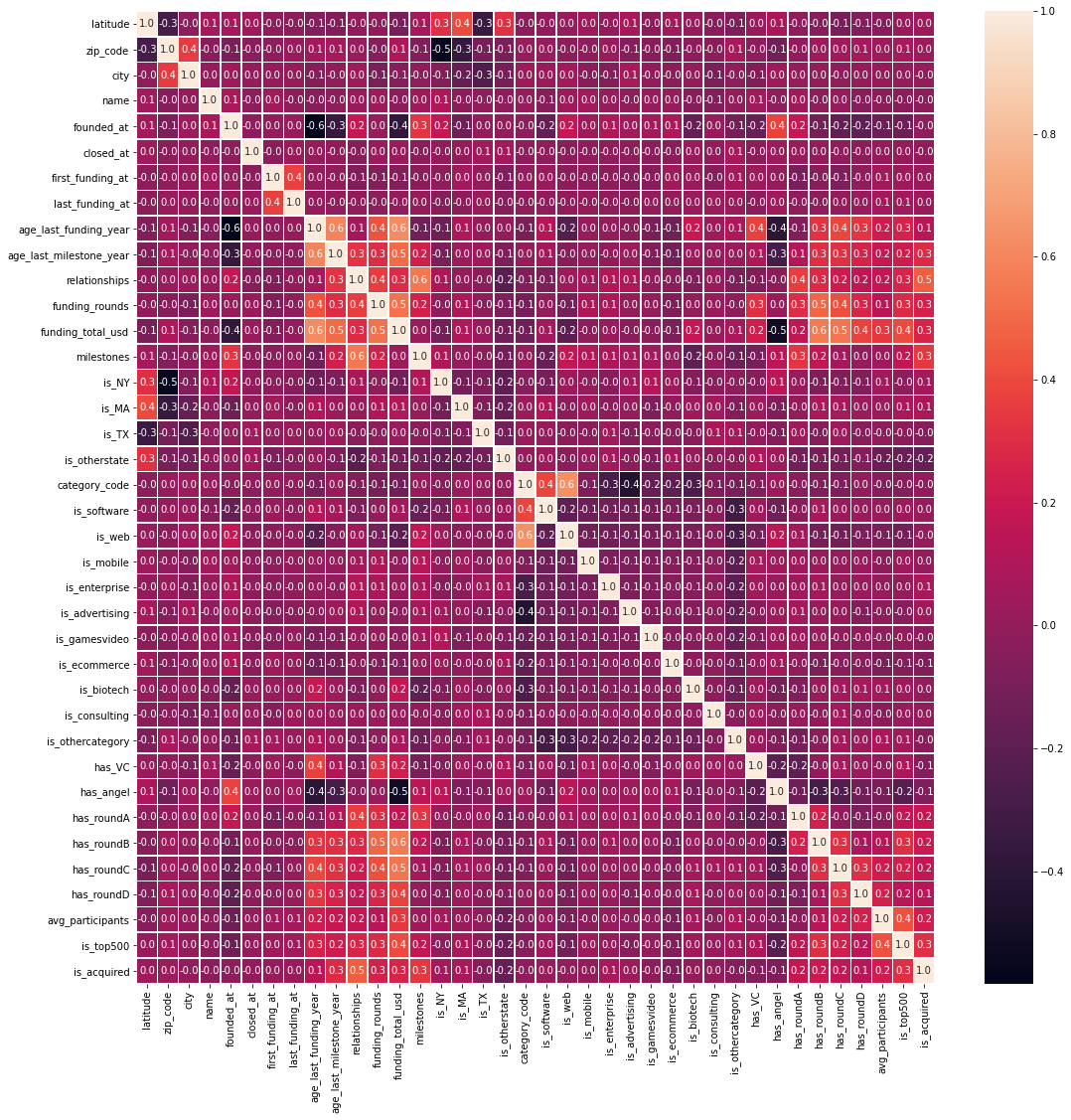


No relation

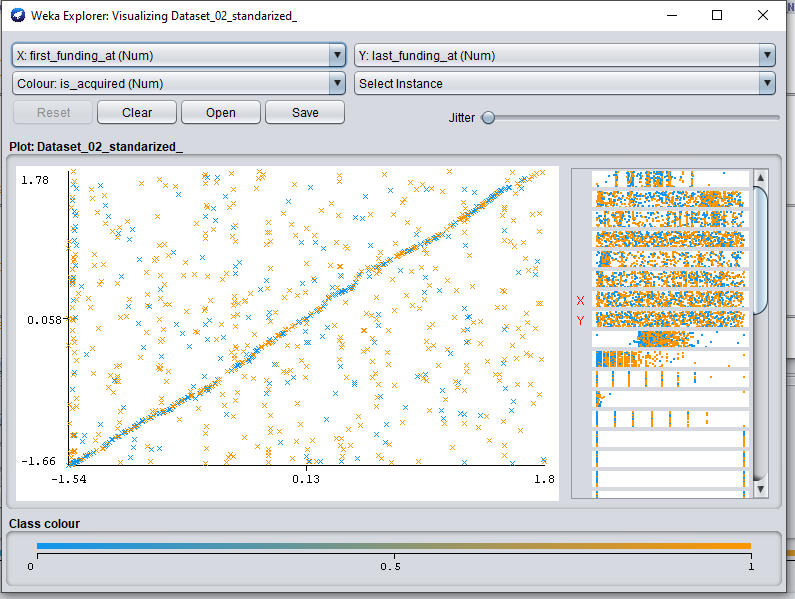


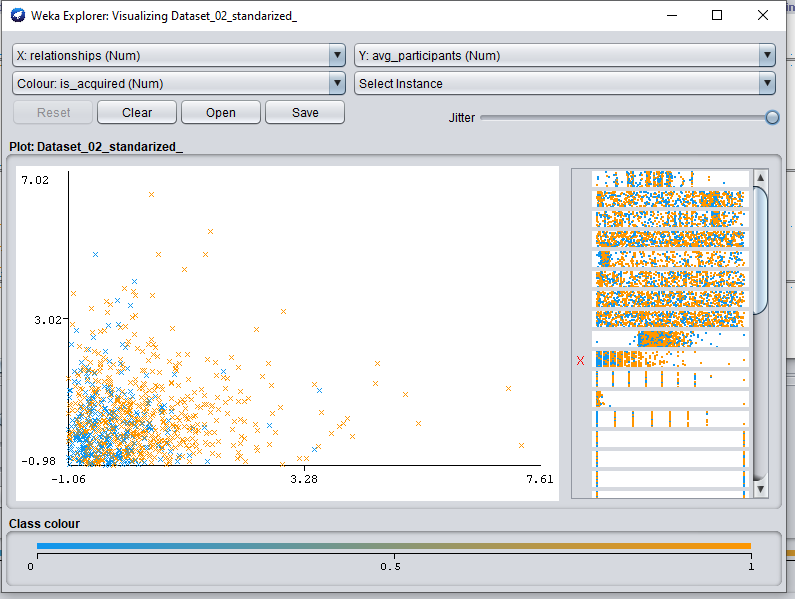
Now using the dataset two.

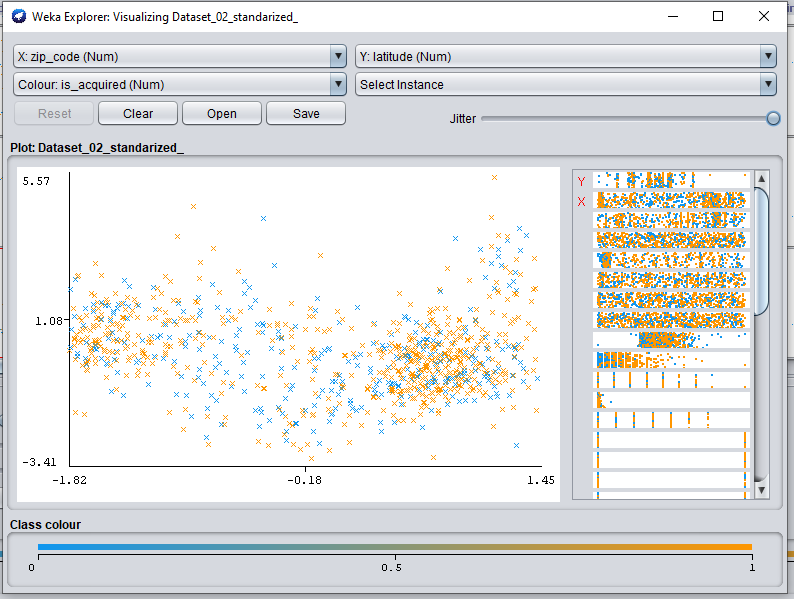
So just for reference lets see tha already computed graph.



We have following co-relations found in this graph.







# Applying classifier algorithms

Now we will apply following classification algorithms multiple times. And each time we will change the test size and notice the difference in accuracy.

## Effect of changing test size: (Features are processed without applying SMOTE and with applying only standardization methods).

### Test 01: For test size=0.2

|  |  |  |
| --- | --- | --- |
|  | test\_size=0.2 | |
|  | Accuracy | Cross Validation Score |
| SGDClassifier | 66.53% | 69.99% (+/- 4.81%) |
| SVC | 79.92% | 76.27% (+/- 2.79%) |
| NuSVC | 79.92% | 74.22% (+/- 5.28%) |
| LinearSVC | 74.06% | 75.63% (+/- 3.13%) |
| KNeighborsClassifier | 72.38% | 66.96% (+/- 4.00%) |
| GaussianNB | 64.44% | 59.89% (+/- 20.59%) |
| RandomForestClassifier | 82.43% | 78.77% (+/- 4.50%) |
| ExtraTreesClassifier | 84.94% | 76.38% (+/- 3.57%) |
| DecisionTreeClassifier | 71.55% | 69.77% (+/- 5.10%) |

### Test 02 :For test size=0.4

|  |  |  |
| --- | --- | --- |
|  | test\_size=0.4 | |
|  | Accuracy | Cross Validation Score |
| SGDClassifier | 64.23% | 69.99% (+/- 2.65%) |
| SVC | 75.73% | 76.27% (+/- 2.79%) |
| NuSVC | 75.73% | 74.22% (+/- 5.28%) |
| LinearSVC | 71.76% | 75.41% (+/- 2.85%) |
| KNeighborsClassifier | 68.20% | 66.96% (+/- 4.00%) |
| GaussianNB | 64.64% | 59.89% (+/- 20.59%) |
| RandomForestClassifier | 79.08% | 78.23% (+/- 2.24%) |
| ExtraTreesClassifier | 80.33% | 75.73% (+/- 3.30%) |
| DecisionTreeClassifier | 74.06% | 69.55% (+/- 5.28%) |

### Test 03 :For test size=0.5

|  |  |  |
| --- | --- | --- |
|  | test\_size=0.5 | |
|  | Accuracy | Cross Validation Score |
| SGDClassifier | 59.80% | 69.12% (+/- 3.95%) |
| SVC | 75.38% | 76.27% (+/- 2.79%) |
| NuSVC | 75.38% | 74.22% (+/- 5.28%) |
| LinearSVC | 72.70% | 75.73% (+/- 2.77%) |
| KNeighborsClassifier | 66.67% | 66.96% (+/- 4.00%) |
| GaussianNB | 63.82% | 59.89% (+/- 20.59%) |
| RandomForestClassifier | 80.23% | 79.52% (+/- 3.44%) |
| ExtraTreesClassifier | 78.56% | 75.95% (+/- 3.37%) |
| DecisionTreeClassifier | 71.52% | 70.09% (+/- 6.11%) |

### Test 04 :For test size=0.7

|  |  |  |
| --- | --- | --- |
|  | test\_size=0.7 | |
|  | Accuracy | Cross Validation Score |
| SGDClassifier | 72.49% | 68.36% (+/- 4.33%) |
| SVC | 70.81% | 76.27% (+/- 2.79%) |
| NuSVC | 70.81% | 74.22% (+/- 5.28%) |
| LinearSVC | 70.81% | 75.73% (+/- 2.68%) |
| KNeighborsClassifier | 61.96% | 66.96% (+/- 4.00%) |
| GaussianNB | 59.09% | 59.89% (+/- 20.59%) |
| RandomForestClassifier | 78.71% | 79.42% (+/- 3.22%) |
| ExtraTreesClassifier | 77.75% | 75.30% (+/- 2.41%) |
| DecisionTreeClassifier | 68.06% | 70.21% (+/- 6.82%) |

### Summary of effect of changing the test size:

Well after analyzing the above tables I have conclude few things about the effect of changing test size on classifiers. Each classifier has shown different behavior. Few classifiers were getting better accuracy and cross fold score as the test size was 0.2 and few showed better results as the partition got around 0.5-0.7. But the **key point** notice was that on **0.2 test size** almost all algorithms were showing better accuracy and better cross validation score. So from this I have concluded that test size affects a lot to the accuracy and cross-validation of algorithm. And if we want to get better results of classifiers for this dataset we have to split test and training data for 0.2

## Effect of providing smoted dataset (Features are standardized before processing for classification)

For this we will just match for only 0.2 test size.

### Without SMOTED dataset we had

|  |  |  |
| --- | --- | --- |
|  | test\_size=0.2 | |
|  | Accuracy | Cross Validation Score |
| SGDClassifier | 66.53% | 69.99% (+/- 4.81%) |
| SVC | 79.92% | 76.27% (+/- 2.79%) |
| NuSVC | 79.92% | 74.22% (+/- 5.28%) |
| LinearSVC | 74.06% | 75.63% (+/- 3.13%) |
| KNeighborsClassifier | 72.38% | 66.96% (+/- 4.00%) |
| GaussianNB | 64.44% | 59.89% (+/- 20.59%) |
| RandomForestClassifier | 82.43% | 78.77% (+/- 4.50%) |
| ExtraTreesClassifier | 84.94% | 76.38% (+/- 3.57%) |
| DecisionTreeClassifier | 71.55% | 69.77% (+/- 5.10%) |

### With SMOTED dataset we had

|  |  |  |
| --- | --- | --- |
|  | test\_size=0.2 | |
|  | Accuracy | Cross Validation Score |
| SGDClassifier | 66.95% | 67.93% (+/- 7.15%) |
| SVC | 79.92% | 76.27% (+/- 2.79%) |
| NuSVC | 79.92% | 74.22% (+/- 5.28%) |
| LinearSVC | 74.06% | 75.73% (+/- 2.77%) |
| KNeighborsClassifier | 72.38% | 66.96% (+/- 4.00%) |
| GaussianNB | 64.44% | 59.89% (+/- 20.59%) |
| RandomForestClassifier | 81.59% | 79.63% (+/- 3.50%) |
| ExtraTreesClassifier | 82.85% | 75.73% (+/- 3.26%) |
| DecisionTreeClassifier | 73.22% | 70.74% (+/- 4.25%) |

### Summary of effect of providing not smoted dataset

Since in this this test we have come across to notice an interesting behavior of smoted dataset and non smoted data set. In sum up this has affected the accuracy; most of algorithms has reduced the accuracy and increased the cross validation score for few alogorithms. Since we about the concept of cross validation score it is basically how well our model will perform on new data until we actually test it. So point which I have noticed here is that when we gave the data which was actually taken from real life dataset (which mean we did not applied any technique like we have done over fitting in above case, to which we call SMOTING.) then this increase the cross validation. So if I conclude this then I can say cross validation is depending on how much dataset is near to reality. And I have found another thing that increasing the number of records does help to train a model which can be used in real life. Because it reduces the cross-validation score (As you can refer to table with smoted dataset results).

## Effect of providing no standardized features (Dataset is SMOTED)

For this we will just match for only 0.5 test size.

### With standardized values.

|  |  |  |
| --- | --- | --- |
|  | test\_size=0.2 | |
|  | Accuracy | Cross Validation Score |
| SGDClassifier | 66.95% | 67.93% (+/- 7.15%) |
| SVC | 79.92% | 76.27% (+/- 2.79%) |
| NuSVC | 79.92% | 74.22% (+/- 5.28%) |
| LinearSVC | 74.06% | 75.73% (+/- 2.77%) |
| KNeighborsClassifier | 72.38% | 66.96% (+/- 4.00%) |
| GaussianNB | 64.44% | 59.89% (+/- 20.59%) |
| RandomForestClassifier | 81.59% | 79.63% (+/- 3.50%) |
| ExtraTreesClassifier | 82.85% | 75.73% (+/- 3.26%) |
| DecisionTreeClassifier | 73.22% | 70.74% (+/- 4.25%) |

### Without standardized values

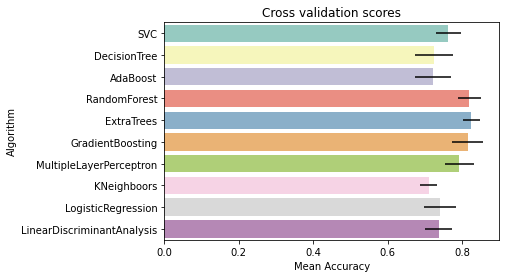
|  |  |  |
| --- | --- | --- |
|  | test\_size=0.2 | |
|  | Accuracy | Cross Validation Score |
| SGDClassifier | 69.87% | 67.06% (+/- 3.95%) |
| SVC | 77.41% | 76.27% (+/- 2.79%) |
| NuSVC | 77.41% | 74.22% (+/- 5.28%) |
| LinearSVC | 74.06% | 75.73% (+/- 2.77%) |
| KNeighborsClassifier | 70.71% | 66.96% (+/- 4.00%) |
| GaussianNB | 64.44% | 59.89% (+/- 20.59%) |
| RandomForestClassifier | 81.17% | 78.44% (+/- 0.71%) |
| ExtraTreesClassifier | 82.85% | 74.43% (+/- 1.65%) |
| DecisionTreeClassifier | 75.31% | 70.10% (+/- 4.24%) |

### Summary of effect of providing not standardized values

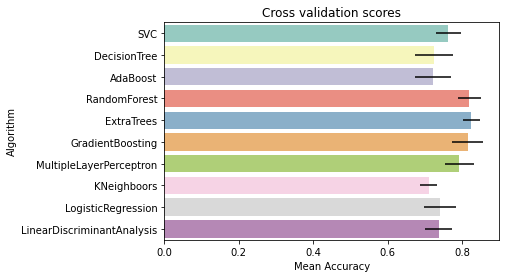
Since in this test I have notice more interesting effects on accuracy of different algorithms. There are three algorithms named: RandomForestClassifier, ExtraTreesClassifier, and DecisionTreeClassifier these are the algorithms which are not affected by non-standardized values. And the rest of algorithms (tested in this experiment) are affected greatly by providing the non-standardized values.

# Comparison of different algorithms with respect to accuracy

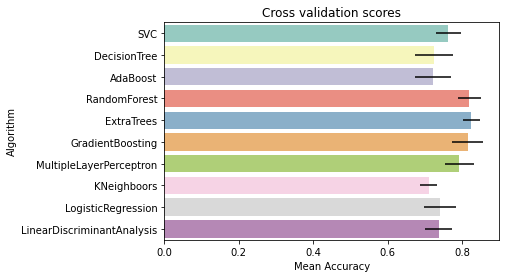
## Test size=0.2



## Test size = 0.4



## Test size=0.5



Summary:

Since we can see Extra tees algorithm gives maximum accuracy in all cases.