|  |
| --- |
| 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? 3](#_Toc72968469)

[Pre-processing 5](#_Toc72968470)

[Dataset 1 (big-mart-sales): 5](#_Toc72968471)

[1. Applying some pre-processing steps (Data set 1): 5](#_Toc72968472)

[1.1 Matching field’s values: 5](#_Toc72968473)

[1.2Filling out missing values: 5](#_Toc72968474)

[1.4 Conversion of string values to numeric values. 6](#_Toc72968475)

[1.5 Conversion of normal values to standardized values. 7](#_Toc72968476)

[1.5Implementation of above stated issues 7](#_Toc72968477)

[Dataset 2 (start-up-success-pridiction): 13](#_Toc72968478)

[2. Applying some pre-processing steps (Data set 2): 13](#_Toc72968479)

[2.2Filling out missing values: 13](#_Toc72968480)

[2.2 Conversion of string values to numeric values. 14](#_Toc72968481)

[1.5 Conversion of normal values to standardized values. 15](#_Toc72968482)

[1.5Implementation of above stated issues 15](#_Toc72968483)

[Data visualization 25](#_Toc72968484)

[Box-Plot 25](#_Toc72968485)

[Summary of using standardized and non-standardized values 28](#_Toc72968486)

[Scatter plot 29](#_Toc72968487)

[Applying classifier algorithms 37](#_Toc72968488)

[Test 01: [Different test-sizes, dataset Is normalized, and dataset is smoted] 37](#_Toc72968489)

[Summary of effect of change in test-size: 38](#_Toc72968490)

[Confusion matrix for best accuracy results of each algorithm 40](#_Toc72968491)

[What is a Confusion Matrix? 40](#_Toc72968492)

[So we will try to analyze following things 42](#_Toc72968493)

[Sensitivity: 42](#_Toc72968494)

[Specificity 42](#_Toc72968495)

[Error rate 42](#_Toc72968496)

[Precision 42](#_Toc72968497)

[Cross-validation (By using python) 42](#_Toc72968498)

[MCC 42](#_Toc72968499)

[Classification results summary and details 43](#_Toc72968500)

# 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://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>

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):

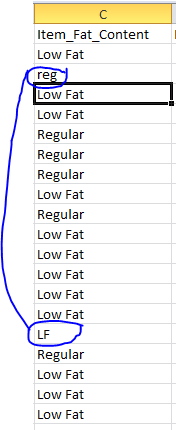
## 1. Applying some pre-processing steps (Data set 1):

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:

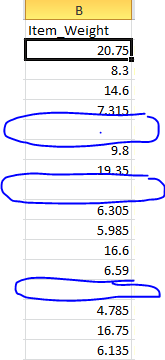
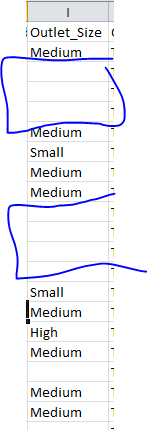
There are few fields in dataset which contain same data with different names, so we need to make the same. For example we have attribute item\_Fat\_Content which contain two labels, Low Fat and Regular, but for representing this same value “LF” and “reg” are used so we need to remove these shortcuts.



### 1.2Filling out missing values:

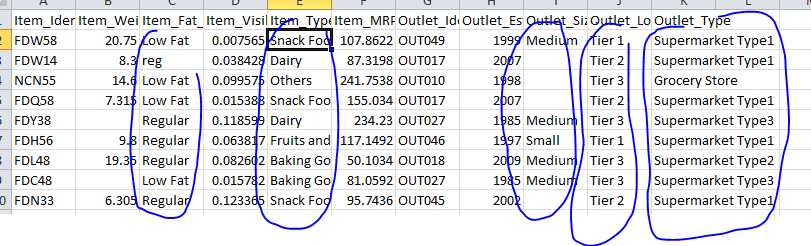
In our dataset we have a bunch of attributes having missing values. And we have to fill them by using well known pre-processing techniques. For example for numeric attribute we have methods like by using median, mean and mode and for ordinal attributes we will use some built in functions of python, like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

For example we have some attributes with missing values.

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

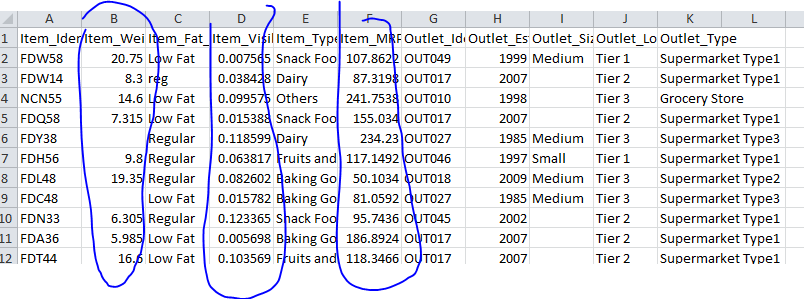
We have some values which are in string form so we need to transform them in numeric form. Since we have following values which need to be in numeric form.



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.

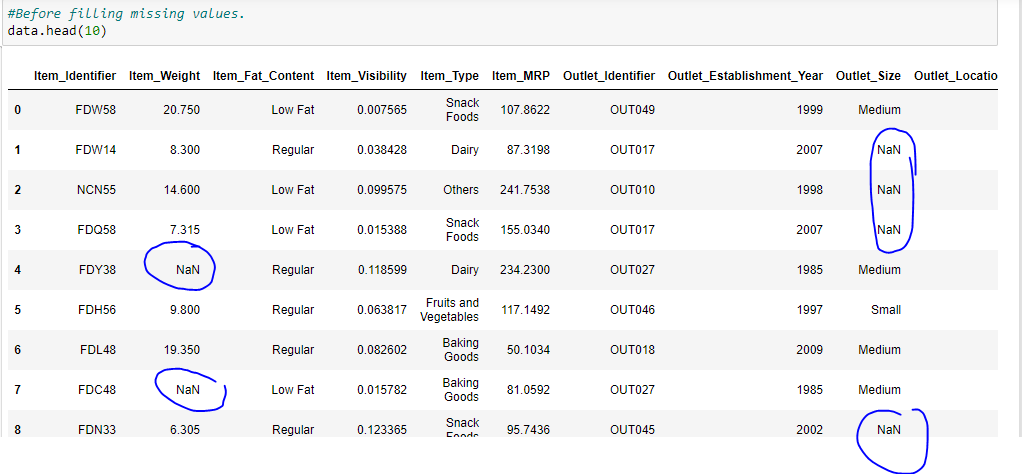


As highlighted values are those one which need to be standardized because this will affect the co-relation graph, which will be drawn later on.

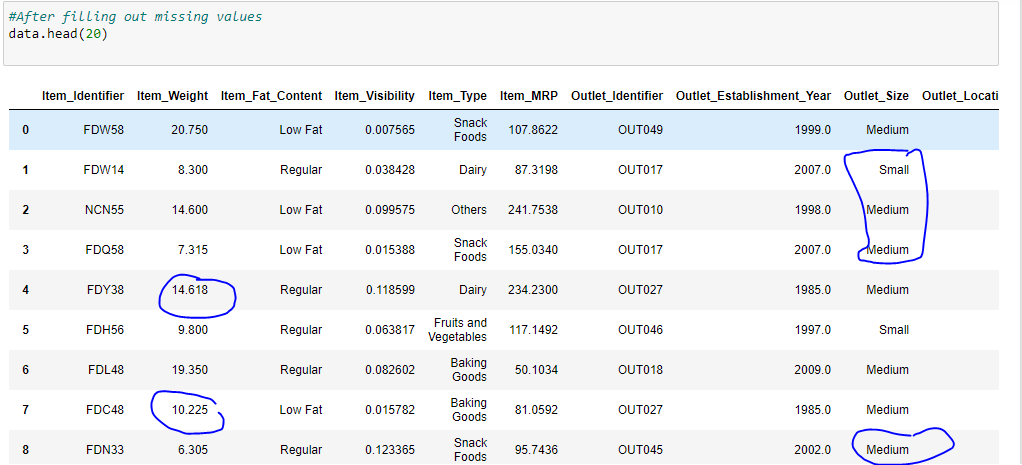
### 1.5Implementation of above stated issues

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.

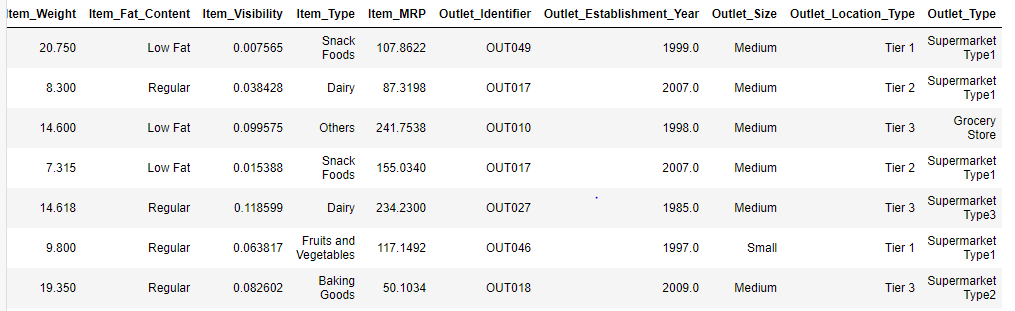
#### Before filling missing values



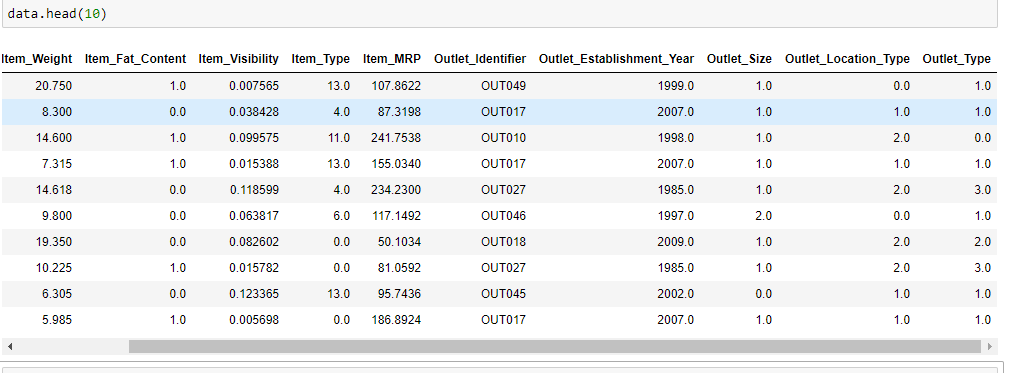
#### After filling missing values



#### Before converting string values to numeric form

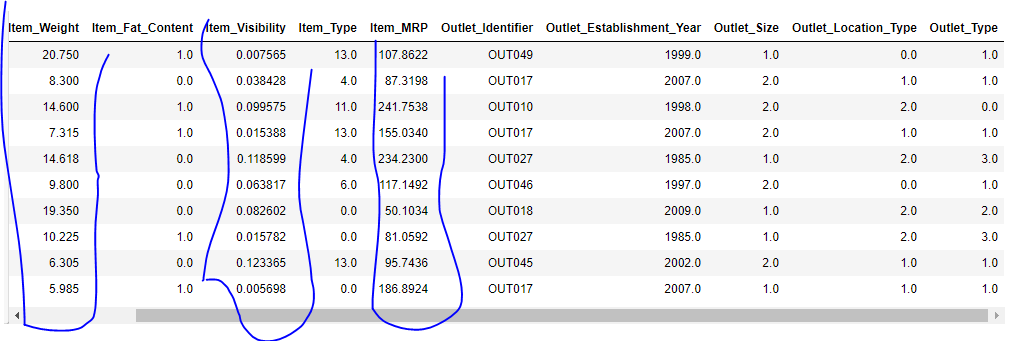


#### After converting string values to numeric form

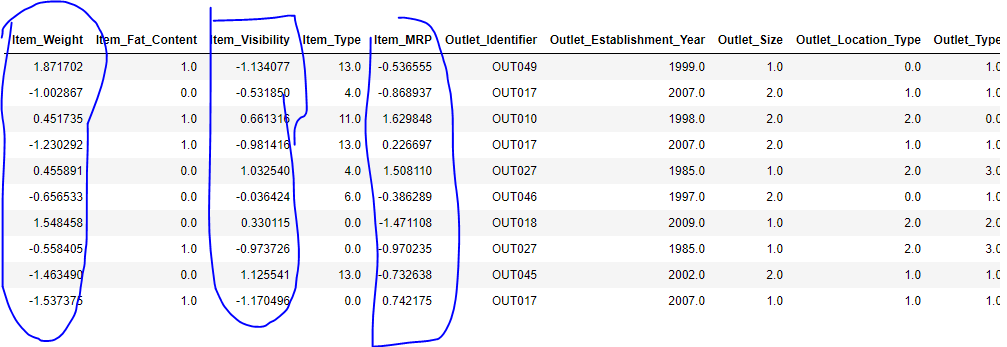


#### Before standardization of values

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> )

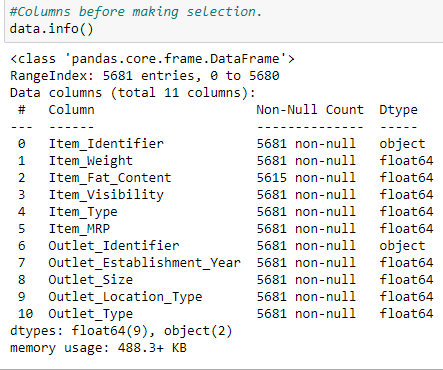


#### After standardization of values



#### 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)

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

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

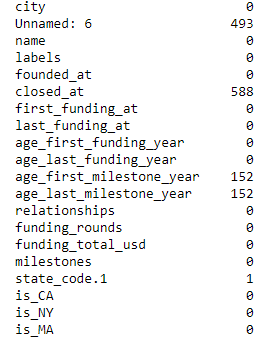
First we need to apply some pre-processing techniques before we process it.

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

### 2.2Filling out missing values:

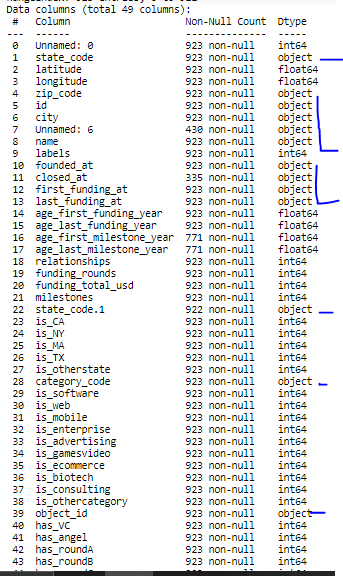
In our dataset we have a bunch of attributes having missing values. And we have to fill them by using well known pre-processing techniques. For example for numeric attribute we have methods like by using median, mean and mode and for ordinal attributes we will use some built in functions of python, like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



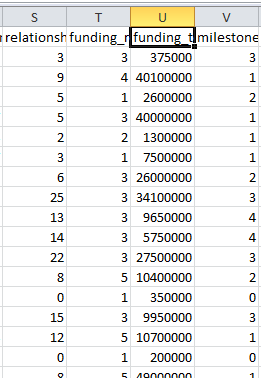
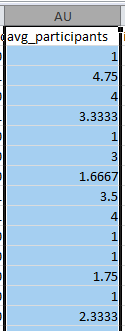
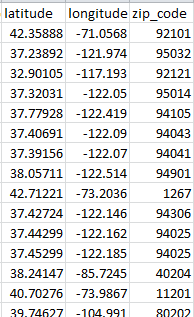
### 2.2 Conversion of string values to numeric values.

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.

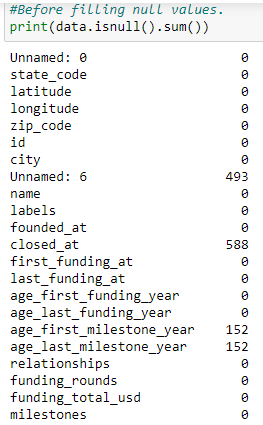
  

As highlighted values are those one which need to be standardized because this will affect the co-relation graph, which will be drawn later on. And we will have some other columns aswell which will be in numeric form after conversion.

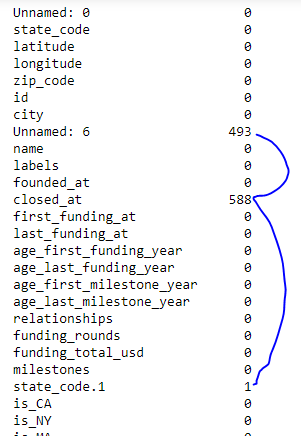
### 1.5Implementation of above stated issues

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.

#### Before filling missing values

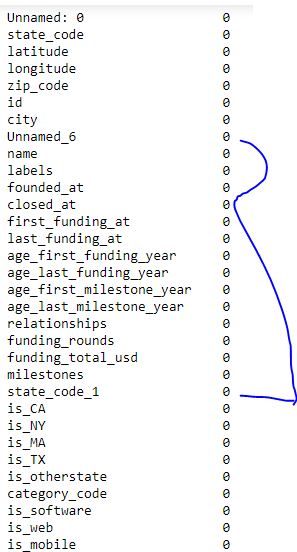


#### After filling missing values



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

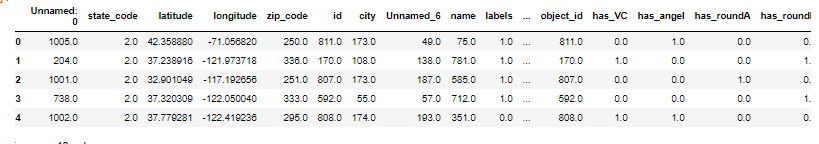
47 is\_top500 923 non-null float64

48 is\_acquired 923 non-null int64

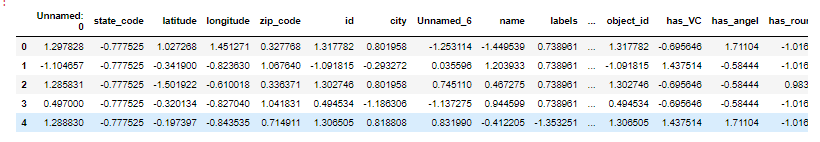
**Now we have successfully converted the string value to numeric form.**

#### 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

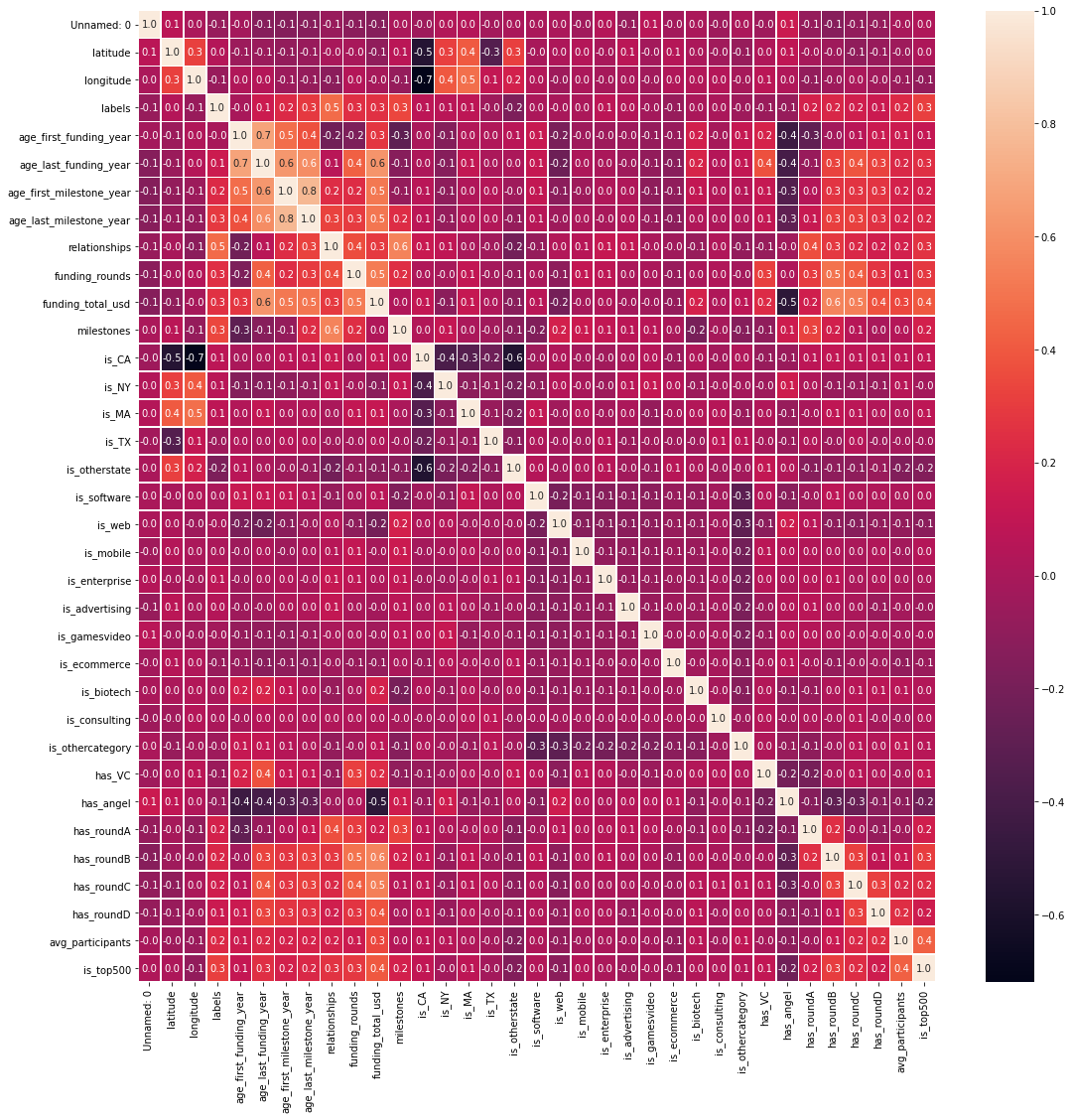


#### Now making selection of features

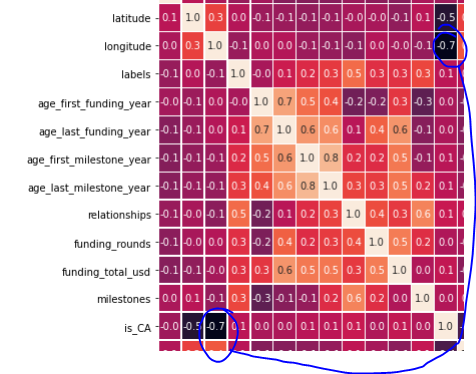
#### Columns before selection

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

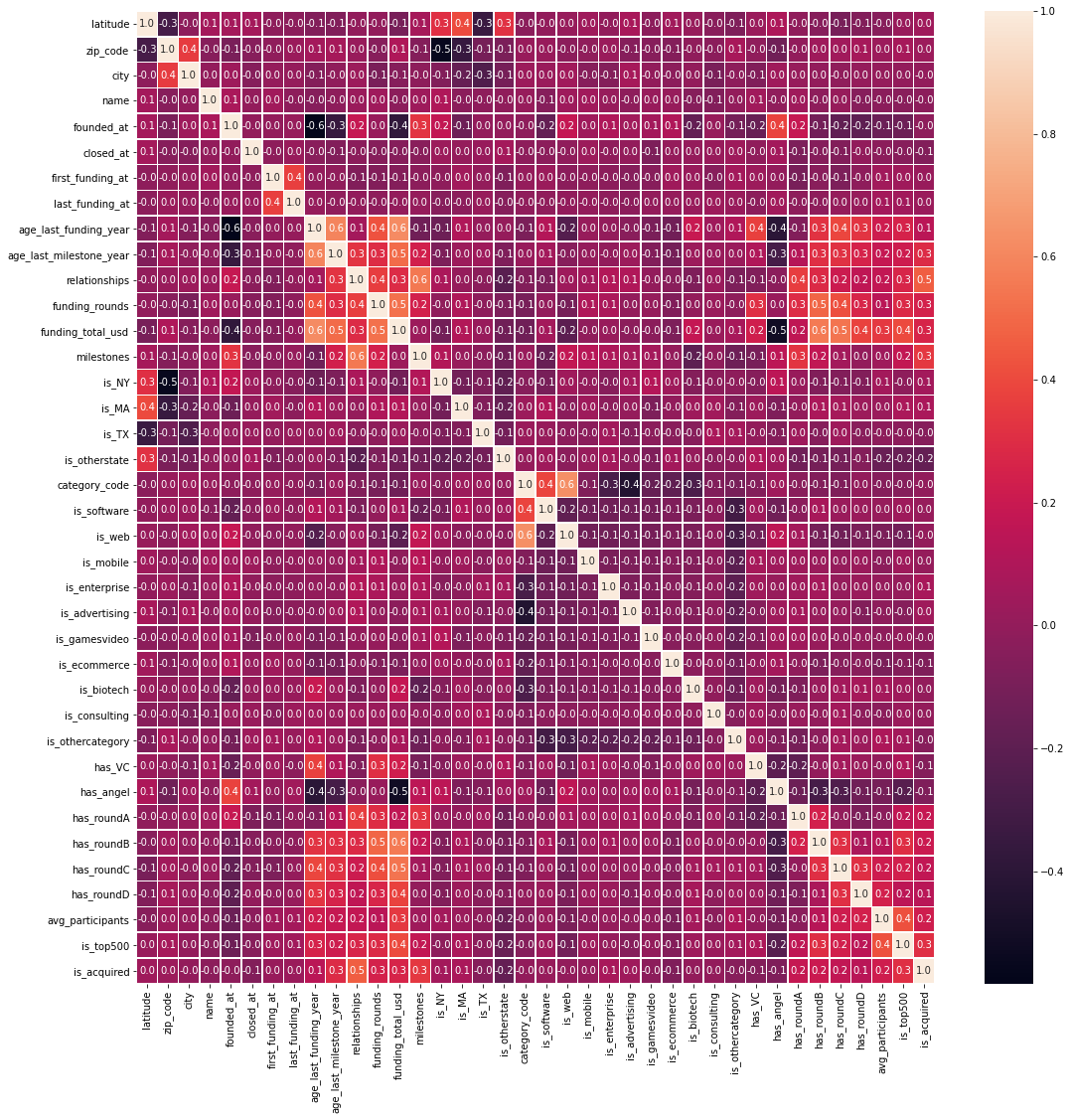
# • labels

# • 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, 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.

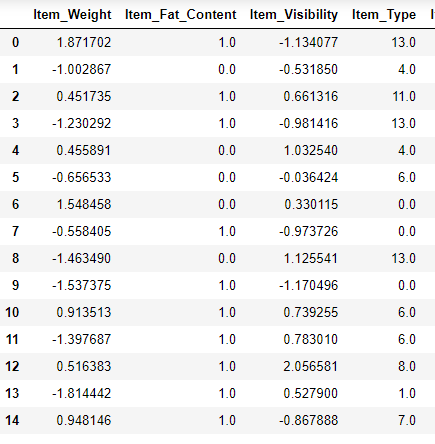
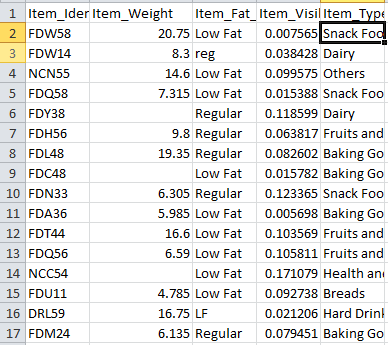
* Detect outlier values.
* Mean tendency of values.
* Symmetry of data.

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.

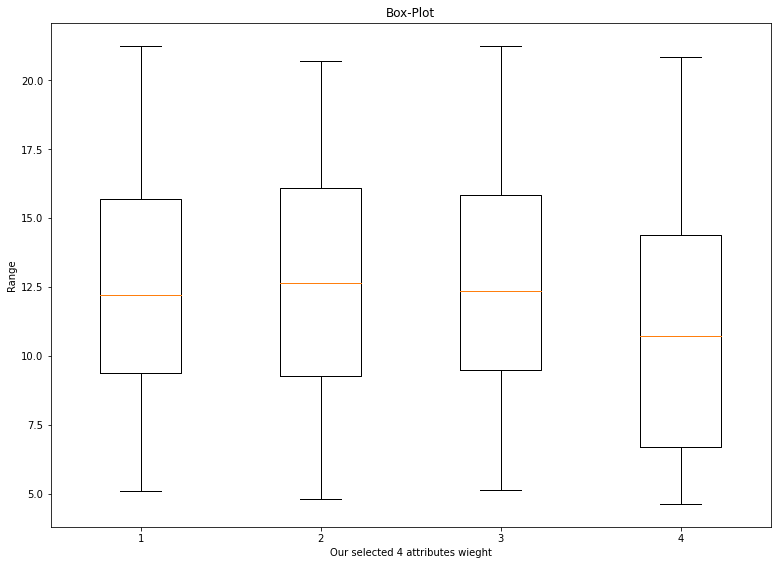
 

**13.0 = Snack Food, 4.0 = Diary, 1.0 = bread and meat=10.0**

For example we are taking 4 items type.

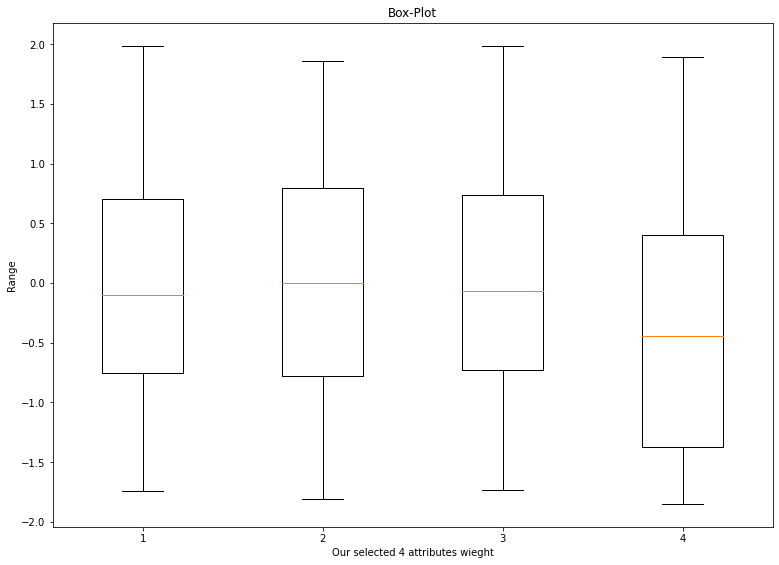
1. Snack foods
2. Dairy
3. Meat
4. Breads

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/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.

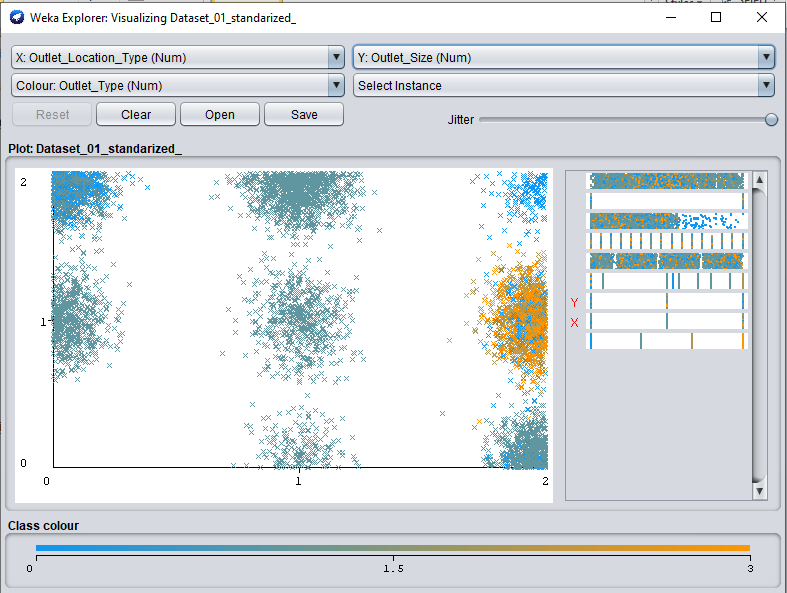


From this above graph we have found only few co-related relations.

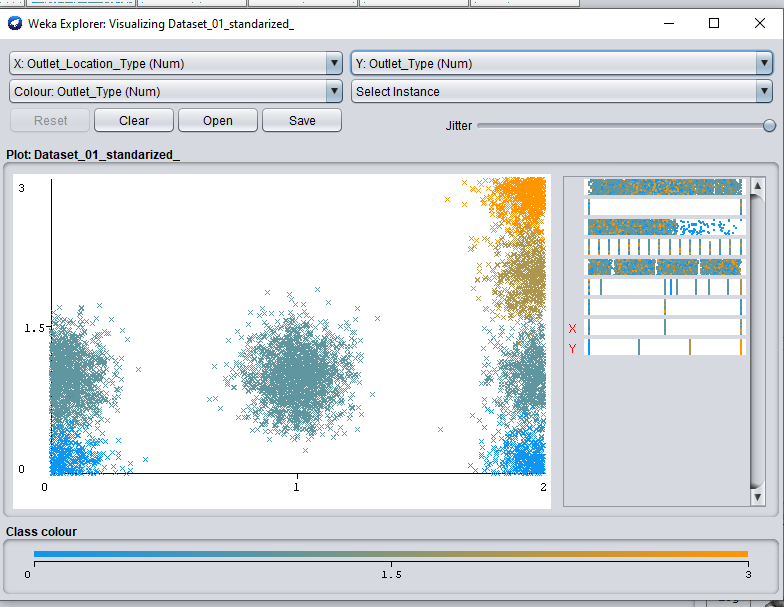
1. Negetive moderate co-relation between Outlet\_size and outlet\_location\_type.
2. Positive moderate co-relation between outlet\_location\_type and outlet\_type.
3. Very weak positive co-relation between item\_fat\_content and item\_type.

Now lets see their scattered plots.

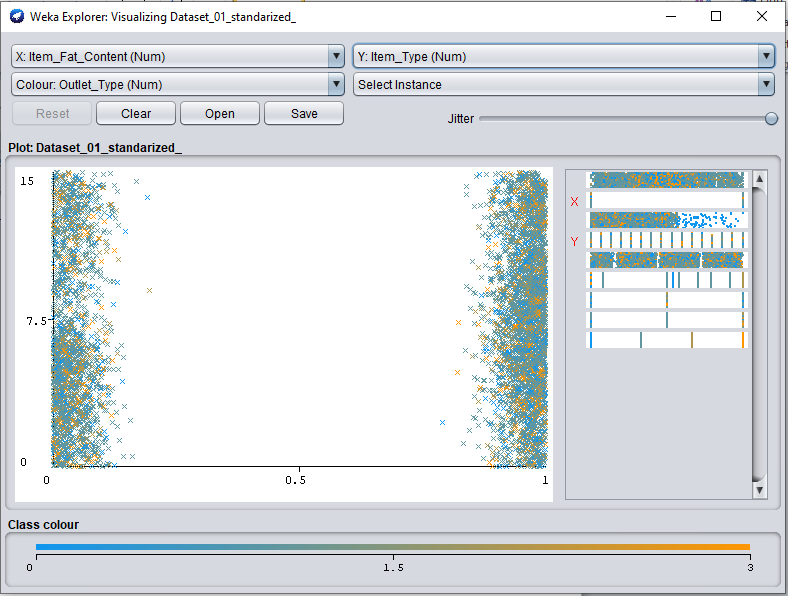
Negetive moderate co-relation between Outlet\_size and outlet\_location\_type.



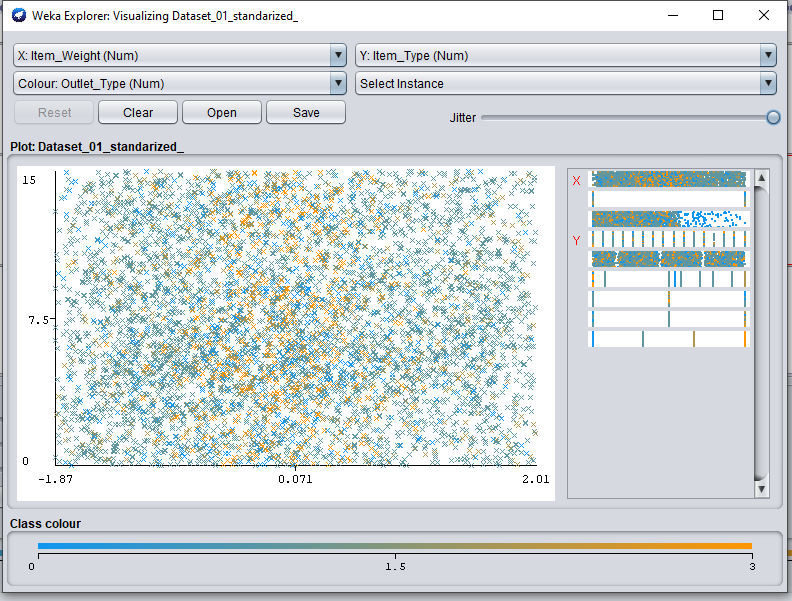
Positive very weak co-relation between outlet\_location\_type and outlet\_type.



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

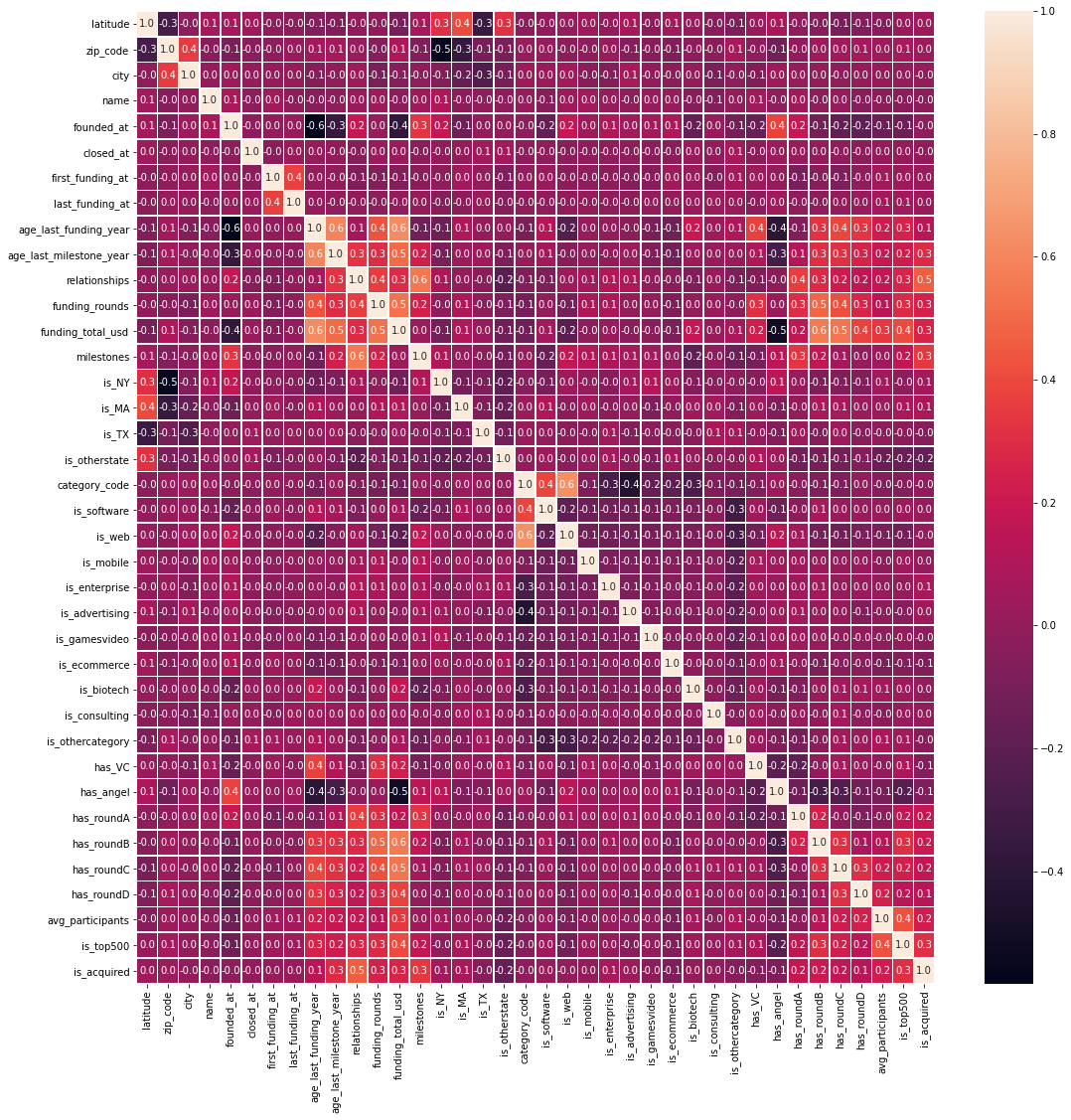


No relation

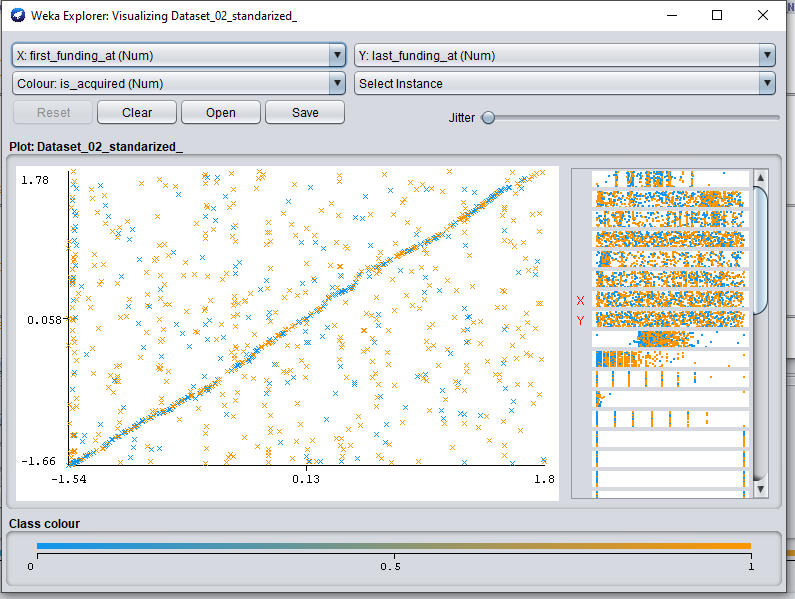


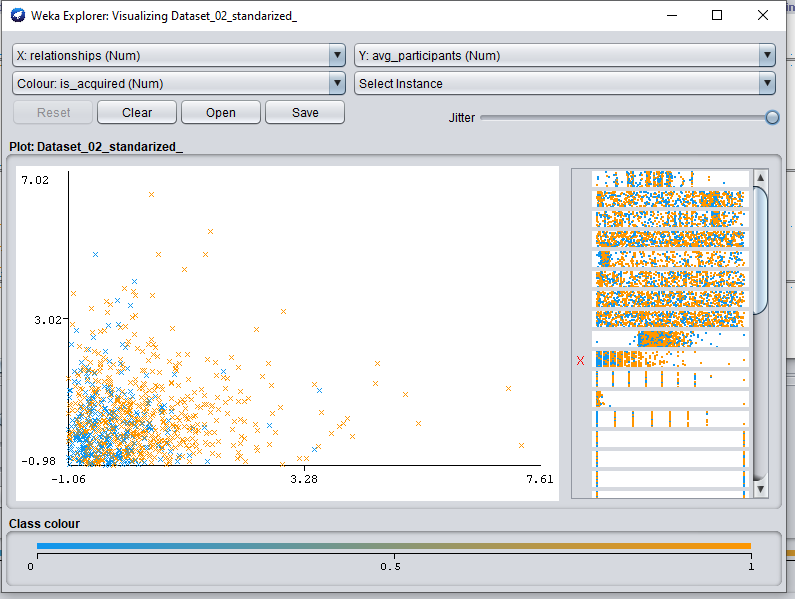
Now using the dataset two.

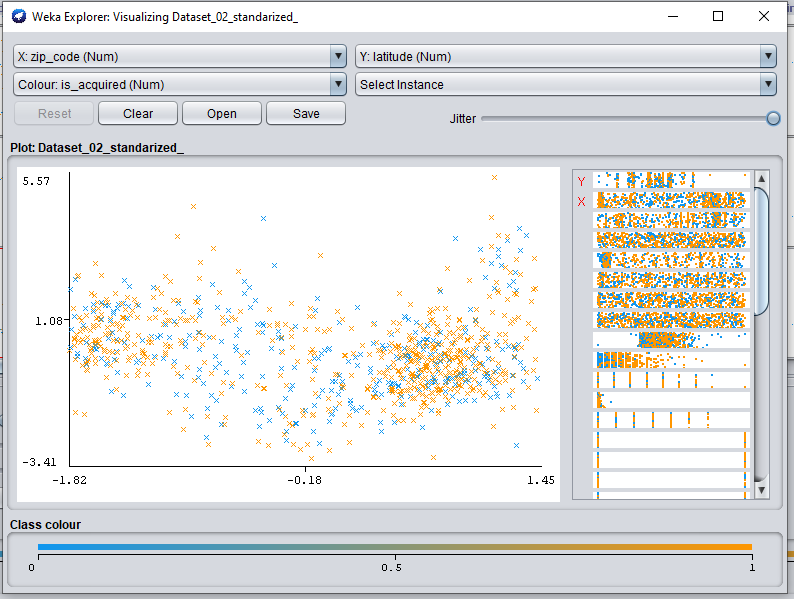
So just for reference lets see tha already computed graph.



We have following co-relations found in this graph.







# Applying classifier algorithms

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

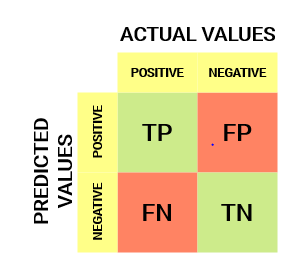
Before we move ahead for analyzing the confution matrix let’s see few definitions which will help us in understanding the confusion matrix.

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**

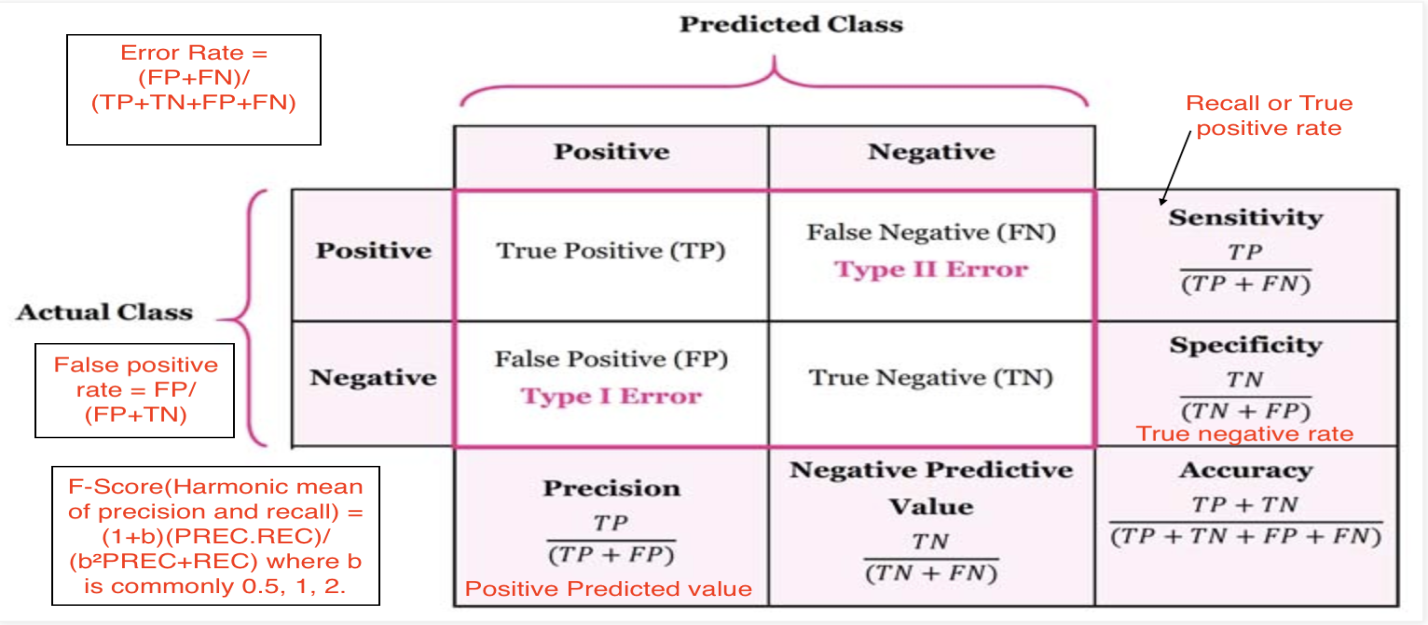
The different values of the Confusion matrix would be as follows:

* 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**

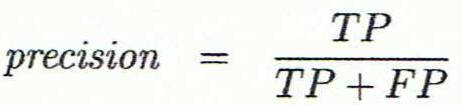
### Specificity

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

### Error rate

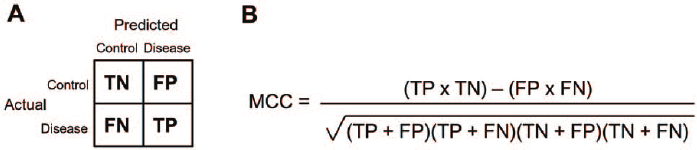
* + *1 –* *accuracy*, or
    - **Error rate = (FP + FN)/All**

### Precision

* + exactness – what % of tuples that the classifier labeled as positive are actually positive
  + 

### Cross-validation (By using python)

### MCC

* + 

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

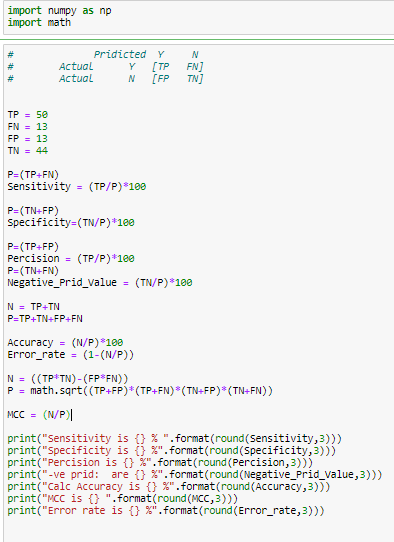
1 : “Acquired ” and 0: “Closed” Number of records while classification 1194

### 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% | con_1 | 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%) |
| GaussianNB | 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%) |
| RandomForestClassifier | 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%) |
| DecisionTreeClassifier | 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>



The End