HIGH VALUE CUSTOMER IDENTIFICATION
UK-BASED ONLINE RETAIL STORE

Priya Johny
DataMining
George Washington University
Date: 27-04-2021

Table of Content

1. Introduction

Objective

The objective of this project is to apply data mining techniques in python language to a real world problem. My area of interest is to explore e-commerce and related industries and therefore have decided to work on a dataset of 'high value customer identification.' Further, applied a data mining algorithm K-means clustering to explore models and gain valuable insights. This model will show the effect that features have on outcome.

Project Overview

Project objective with the dataset obtained is to find 'significant customers' for the business who make high purchases of their favorite products. Dataset used is obtained from a UK-based online retail store on high value customer identification (customers with high purchases.) This dataset contains transactions occurring between November-2016 to December-2017 for the UK based online retail store.

- Data mining algorithm used is K-means clustering and it is in standard form to segment customers into groups. This will help find the number of customers who are highly valued as well as helps find an algorithm that can give maximum accuracy.
- The python libraries used to implement the network in this project are NumPy, Pandas, Seaborn, Matplotlib, Sklearn and Scipy.

2. Description

Project Role

(a.) Acquire Dataset

Acquiring the right dataset for a problem statement is essential. Understanding the problem, domain, and data is extremely important for building high performing models. Here, first individually we worked on exploring the dataset from various sources and noted observations and analysed which algorithms will suit the acquired dataset. Sources used are Kaggle, SQLBELLE and Dataset Search. In this case, the dataset is an unsupervised learning where the outcome variable is unknown to us.

(b.) Understanding Clustering Types

Some of the clustering types are as follows:

- Hierarchical clustering
- K-Means clustering
- KNN (K-nearest neighbors)
- Principal Component Analysis
- Singular Value Decomposition
- Independent Component Analysis

K-means clustering algorithm is used for this dataset to cluster customers into clusters such as high-value customers, regular-customers and irregular customers for a loyalty program. Initially, the desired number of clusters are selected. In this clustering method, data points need to be clustered into k groups. A larger k means smaller groups with more granularity in the same way , a lower k means larger groups with less granularity.

The output of the algorithm is a group of "labels." It assigns data points to one of the k groups. In k-means clustering, each group is defined by creating a centroid for each group. The centroids are like the heart of the cluster, which captures the points closest to them and adds them to the cluster. Through data preprocessing, we will be able to visualize the available features and can decide which features can be used for the problem statement.

(c.) Introduce the Data

Here, “UK-high value customers identification,” is used as our dataset. A quick visual overview of the dataset is done to decide if the dataset is noisy. In this case, the acquired dataset is noisy and hence needs to be cleaned. The next step is to determine if the dataset can be used as a classification or as a regression problem.

(d.) Basic Data Cleaning

1. **Dealing with data types**

Understanding the data types of the dataset is important to decide if the data type can be handled by the model. There are three main data types:

(a.) Numeric(deals with numerical values e.g. age, height, income etc)

(b.) Categorical (e.g. gender, nationality) and

(c.) Ordinal (e.g high/medium/low)

It is important to understand that models can only handle numeric features and therefore categorical or ordinal should be converted to numeric form. This can be done either by creating dummy features or label encoded In the set of dummy features, 1 indicates that the observation belongs to that category. In pandas, pandas.get_dummies() can be used. Pandas makes pre-modelling workflow easier.

2. **Handling Missing Data**

Data models cannot handle missing values and therefore the easiest way is to get rid of the missing values is by removing them if the dataset is fairly large else use imputation to replace missing values as it can cause issues and potential biases. Imputation is nothing but replacing missing values with either mean,median,or highest frequency value based on the feature.

3. **Whitespace/Symbols**

When the data is of object type it can contain contain whitespace, symbol(*^#) and more that cause difficulty for the model and hence it is good to replace with no space or underscore. Feature ‘description’ contains ‘white spaces”, therefore used underscore.

4. **Statistical Information**

To get to know the statistical relationship of the various features, more information and count of unique values of individual features or duplicates, these can be viewed with the help of pandas functions like data_frame.describe() or data_frame.info() or data_frame.value counts() or data_frame.()

5. **Slicing**

Once the dataset is explored, we get familiar with what we need to find out and therefore based on requirements we can locate and delete rows or columns

(e.) More Data Exploration

1. **Inputs**

Inputs referring to the independent variable(also known as features) are used as predictors. Check which features can be used to determine the outcome. We can find features such as ‘InvoiceNo’, ‘CustomerId’, ‘Quantity’, ‘UnitPrice’ to be very essential therefore using inputs.

2. **Unique values**

If the dataset contains categorical features then it is necessary to determine which categorical values need to be changed to numeric values. This can be done with the help of the unique() function.

3. **Outputs**

Outputs are the dependent variable(or the outcome) is the target variable for prediction. Here, we do not have a specific feature as outcome variable. We leave it to the model to make predictions.

Using pandas library, explored the shape, data types, summary statistics and then cleaned the dataset that consisted of one null column, whitespace and symbols, missing values in the column named ‘Description’ and ‘CustomerId’.

4. **Analysing**

Once all the data exploration is done along with the help of more visualisation(EDA), we can say ‘UnitPrice’, ‘CustomerId’, ‘Quantity’ and ‘InvoiceNo’ to be the most important features for our K-means modelling. Given a set of observations ($x_1, x_2, \ldots, x_n$), where each observation is a d-dimensional real vector, k-means clustering aims to partition the n observations into k ($\leq$ n) sets S = {$S_1, S_2, \ldots, S_k$} so as to minimize the within-cluster sum of squares (WCSS) (i.e. variance).

Mathematically we find,

$$\boxed{\underset{\mathbf{S}}{\arg\min} \sum_{i=1}^{k} \sum_{\mathbf{x} \in S_i} \|\mathbf{x} - \boldsymbol{\mu}_i\|^2 = \underset{\mathbf{S}}{\arg\min} \sum_{i=1}^{k} |S_i| \operatorname{Var} S_i}$$

Where, μ_i is the mean of points in S_i. Unsupervised algorithms such as the one we will be using for this project, K-means, make inferences from datasets using only input vectors without referring to known, or labelled, outcomes.

3. Code-Work Description

I. Introduce Data

Here we will be using python libraries such as numpy, pandas, matplotlib, seaborn scipy and sklearn.

1. Import Python Libraries

>
```
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
import warnings
import numpy as np
from sklearn.cluster import KMeans
from sklearn.preprocessing import MinMaxScaler
warnings.filterwarnings('ignore')
```

2. Read Dataset

>
```
e_data=pd.read_csv('Ecommerce.csv',encoding='unicode_escape',skipinitialspace = True)
```

>
```
print("The original dataset has", len(e_data), "observations and", len(e_data.columns), "variables/features. \n")
```

>
```
print("Name of all the variables:")
print(e_data.columns, '\n')
```

>
```
print(e_data.head())
print(e_data.tail())
```

3. Label Features Names

>

```
e_data.columns=['InvoiceNo','StockCode','Description','Quan
tity','InvoiceDate','UnitPrice','CustomerID','Country', '
']
```

4. View Statistics

>

```
e_data.info()
```

>

```
print(df.describe())
```

5. Check Missing Values

>

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

6. Unique Values

>

```
print(df.nunique())
```

>

```
for col_name in df.columns:
   if df[col_name].dtype == 'object':
       unique_cat = len(df[col_name].unique())
       print("categorical feature '{col_name}' has {unique_cat}
       unique categories".format
        (col_name=col_name,unique_cat=unique_cat))
```

II. Basic Data Cleaning

1. Drop Last Column

>

```
df = e_data.drop([' '], axis=1, inplace=True)
```

2. Align

>

```
dfStyler=df.style.set_properties(subset=['StockCode'],**{'tex
t-align': 'left'})
```

3. Replace whitespace

>
```
new_data.replace(' ', '_', regex=True,inplace=True)
```

4. Drop missing values

>
```
df.dropna(axis = 0, inplace = True)
```

5. Drop duplicates

>
```
df = df.drop_duplicates(subset =['InvoiceNo', 'CustomerID',
'Description', 'Quantity'], keep = 'first')
```

6. Change datatype

>
```
df['CustomerID'] = df['CustomerID'].astype(int)
```

III. Creating and Expanding Variables

>
```
df['TotalExpense'] = df['Quantity'] * df['UnitPrice']
df['Year'] = df['InvoiceDate'].dt.year
df['Month'] = df['InvoiceDate'].dt.month
df['Day'] = df['InvoiceDate'].dt.day
df.drop(['InvoiceDate'], axis=1, inplace=True)
```

III. Checking the Cleaned Dataset

>
```
print("Number of data points in the final cleaned
dataset:", len(df))
```

>
```
print(e_data.head())
```

III. Basic Gui Trial

>

```
#import tkinter packages
import tkinter as tk
from tkinter import filedialog
from pandas import DataFrame
```

>

```
#read df ->cleaned dataset
df = DataFrame(df)
root = tk.Tk()
```

>

```
GUI = tk.Canvas(root, width=300, height=300, bg='black',
relief='raised')
GUI.pack()
```

>

```
#exportCSV() function is used to convert existing to a new csv
file
def exportCSV():
   global df
   export_file_path =
filedialog.asksaveasfilename(defaultextension='.csv')
   df.to_csv(export_file_path, index=False, header=True)
#button
Button_CSV = tk.Button(text='Export CSV', command=exportCSV,
bg='blue', fg='black',
                            font=('helvetica', 12, 'bold'))
GUI.create_window(150, 150, window=Button_CSV)
root.mainloop()
```

4. Results

Screenshots of Outputs

1. Import Python Libraries

We begin by importing the required python libraries such as pandas, matplotlib, seaborn, numpy, sklearn.

From which the data cleaning mainly uses pandas and the rest are used for visualization and modelling.

CODE:

```
#python libraries

import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
import warnings
import numpy as np
from sklearn.cluster import KMeans
from sklearn.preprocessing import MinMaxScaler
warnings.filterwarnings('ignore')
```

2. Read Dataset

CODE:

```
#%%------------------------------------------------------------------
e_data = pd.read_csv('Ecommerce.csv',encoding='unicode_escape',skipinitialspace = True)
#print(e_data.head(4))
print("The original dataset has", len(e_data), "observations and", len(e_data.columns), "variables/features. \n")
print("Name of all the variables:")
print(e_data.columns, '\n')

e_data.columns=['InvoiceNo',
                'StockCode',
                'Description',
                'Quantity',
                'InvoiceDate',
                'UnitPrice',
                'CustomerID',
                'Country', ' ']
print(e_data.columns)
```

Result:

```
The original dataset has 541909 observations and 9 variables/features.

Name of all the variables:
Index(['InvoiceNo', 'StockCode', 'Description', 'Quantity', 'InvoiceDate',
       'UnitPrice', 'CustomerID', 'Country', 'Unnamed: 8'],
      dtype='object')

Index(['InvoiceNo', 'StockCode', 'Description', 'Quantity', 'InvoiceDate',
       'UnitPrice', 'CustomerID', 'Country', ' '],
      dtype='object')
```

CODE:

```
print(df.head())
print(df.tail())
```

Result:

```
Exploring the head and tail of the dataset

Head:

  InvoiceNo StockCode  ...         Country  Unnamed: 8
0    536365    85123A  ...  United Kingdom         NaN
1    536365     71053  ...  United Kingdom         NaN
2    536365    84406B  ...  United Kingdom         NaN
3    536365    84029G  ...  United Kingdom         NaN
4    536365    84029E  ...  United Kingdom         NaN

[5 rows x 9 columns]
```

Result:

```
Tail:

       InvoiceNo StockCode  ... Country  Unnamed: 8
541904    581587     22613  ...  France         NaN
541905    581587     22899  ...  France         NaN
541906    581587     23254  ...  France         NaN
541907    581587     23255  ...  France         NaN
541908    581587     22138  ...  France         NaN

[5 rows x 9 columns]
```

4. View Statistics

CODE:

```
#%%
print("Datatype of all the variables:")
df.info()
```

Result:

```
Datatype of all the variables:

<class 'pandas.core.frame.DataFrame'>
RangeIndex: 541909 entries, 0 to 541908
Data columns (total 9 columns):
 #   Column       Non-Null Count   Dtype
---  ------       --------------   -----
 0   InvoiceNo    541909 non-null  object
 1   StockCode    541909 non-null  object
 2   Description  540455 non-null  object
 3   Quantity     541909 non-null  int64
 4   InvoiceDate  541909 non-null  datetime64[ns]
 5   UnitPrice    541909 non-null  float64
 6   CustomerID   406829 non-null  float64
 7   Country      541909 non-null  object
 8   Unnamed: 8   0 non-null       float64
dtypes: datetime64[ns](1), float64(3), int64(1), object(4)
memory usage: 37.2+ MB
4 of the variables are object type, quantity variable is in integer type,
 date is in datetime64 and rest are in float datatype
# --------------------------------------------------
```

Code:

```
#%%
print("Statistical Summary :")
print(df.describe())
```

Result:

```
# --------------------------------------------------
Statistical Summary :

            Quantity      UnitPrice     CustomerID  Unnamed: 8
count  541909.000000  541909.000000  406829.000000         0.0
mean        9.552250       4.611114   15287.690570         NaN
std       218.081158      96.759853    1713.600303         NaN
min    -80995.000000  -11062.060000   12346.000000         NaN
25%         1.000000       1.250000   13953.000000         NaN
50%         3.000000       2.080000   15152.000000         NaN
75%        10.000000       4.130000   16791.000000         NaN
max     80995.000000   38970.000000   18287.000000         NaN
Statistical summary give us an overview of the statistics such as mean,
 count,minumum, maximum,percentiles and
 standard deviation values of each variable.
# --------------------------------------------------
```

5. Check Missing Values

CODE:

```
#%%
print("Number of null data-points in each variable:")
print(df.isnull().sum())

print('#', 50*"-")
```

Result:

```
# --------------------------------------------------
Number of null data-points in each variable:

InvoiceNo           0
StockCode           0
Description      1454
Quantity            0
InvoiceDate         0
UnitPrice           0
CustomerID     135080
Country             0
Unnamed: 8     541909
dtype: int64
Variable description has over 1454 null values and
 CustomerId has 135080 null values.
```

6. Unique Values

CODE:

```
#%%
print("Number of unique data-points in each variable:")
print(df.nunique())

#unique values of categorical in case-of large datasets
for col_name in df.columns:
    if df[col_name].dtype == 'object':
        unique_cat = len(df[col_name].unique())
        print("categorical feature '{col_name}' has {unique_cat}  unique categories".format(col_name=col_name,unique_cat=unique_cat))

print('#',50*"-")
```

Result:

```
# --------------------------------------------------
Number of unique data-points in each variable:

InvoiceNo      25900
StockCode       4070
Description     4223
Quantity         722
InvoiceDate      305
UnitPrice       1630
CustomerID      4372
Country           38
Unnamed: 8         0
dtype: int64
```

Result:

```
The function nunique() gives us the unique values present in each variable and
 we can separate based on single variable.
This unique values help us group values with least unique values together in a single group
Display of unique values of type categorical in case-of large datasets

categorical feature 'InvoiceNo' has 25900  unique categories
categorical feature 'StockCode' has 4070  unique categories
categorical feature 'Description' has 4224  unique categories
categorical feature 'Country' has 38  unique categories
# --------------------------------------------------
```

II. Basic Data Cleaning

1. Drop Last Column

CODE:

```
print("Data Cleaning \n")

#%%
print("Dropping the variable Unnamed: 8")
df.drop([' '], axis=1, inplace=True)
```

Result:

```
# ---------------------------------------------
Data Cleaning

Dropping the variable Unnamed: 8
```

2. Align

CODE:

```
dfStyler=df.style.set_properties(subset=['StockCode'],**{'text-align': 'left'})
```

3. Replace whitespace

(optional, in our data modelling we have not used this feature.)

CODE:

```
#%%
new_data.replace(' ', '_', regex=True,inplace=True)
print(new_data['Description'])
```

Result:

```
0         WHITE_HANGING_HEART_T-LIGHT_HOLDER
1                        WHITE_METAL_LANTERN
2             CREAM_CUPID_HEARTS_COAT_HANGER
3        KNITTED_UNION_FLAG_HOT_WATER_BOTTLE
4             RED_WOOLLY_HOTTIE_WHITE_HEART.
                         ...
541904           PACK_OF_20_SPACEBOY_NAPKINS
541905          CHILDREN'S_APRON_DOLLY_GIRL_
541906         CHILDRENS_CUTLERY_DOLLY_GIRL_
541907       CHILDRENS_CUTLERY_CIRCUS_PARADE
541908         BAKING_SET_9_PIECE_RETROSPOT_
Name: Description, Length: 541909, dtype: object
```

4. Drop missing values

CODE:

```
#%%
print("Dropping rows with missing/na values")
df.dropna(axis = 0, inplace = True)
```

Result:

```
Dropping the variable Unnamed: 8

  InvoiceNo StockCode  ... CustomerID         Country
0    536365    85123A  ...    17850.0  United Kingdom
1    536365     71053  ...    17850.0  United Kingdom
2    536365    84406B  ...    17850.0  United Kingdom
3    536365    84029G  ...    17850.0  United Kingdom
4    536365    84029E  ...    17850.0  United Kingdom

[5 rows x 8 columns]
Dropped variable Unnamed: 8
```

5. Drop duplicates

CODE:

```
#%%
print("Checking for duplicates \n")
df = df.drop_duplicates(subset =['InvoiceNo', 'CustomerID', 'Description', 'Quantity'], keep = 'first')
```

Result:

```
# --------------------------------------------------
Changing the datatype of CustomerID to int
# --------------------------------------------------
Checking for duplicates

<class 'pandas.core.frame.DataFrame'>
Int64Index: 401472 entries, 0 to 541908
Data columns (total 11 columns):
 #   Column         Non-Null Count   Dtype
---  ------         --------------   -----
 0   InvoiceNo      401472 non-null  object
 1   StockCode      401472 non-null  object
 2   Description    401472 non-null  object
 3   Quantity       401472 non-null  int64
 4   UnitPrice      401472 non-null  float64
 5   CustomerID     401472 non-null  int64
 6   Country        401472 non-null  object
 7   Year           401472 non-null  int64
 8   Month          401472 non-null  int64
 9   Day            401472 non-null  int64
 10  TotalExpense   401472 non-null  float64
dtypes: float64(2), int64(5), object(4)
memory usage: 36.8+ MB
None
```

6. Change datatype

CODE:

```
#%%
print("Changing the datatype of CustomerID to int")
df['CustomerID'] = df['CustomerID'].astype(int)
print('#',50*"-")
```

III. Creating and Expanding Variables

Year, Month and Day columns are created. This will help in the plotting of graphs and will help in drawing insights purchases made on which day, month and year.

CODE:

```
#%%
print("Extracting year, month and date from the InvoiceDate variable")
df['Year'] = df['InvoiceDate'].dt.year
df['Month'] = df['InvoiceDate'].dt.month
df['Day'] = df['InvoiceDate'].dt.day
df.drop(['InvoiceDate'], axis=1, inplace=True)
```

Result:

```
Extracting year, month and date from the InvoiceDate variable to

  InvoiceNo StockCode                          Description  ...  Year  Month  Day
0    536365    85123A   WHITE HANGING HEART T-LIGHT HOLDER  ...  2016     11   29
1    536365     71053                  WHITE METAL LANTERN  ...  2016     11   29
2    536365    84406B       CREAM CUPID HEARTS COAT HANGER  ...  2016     11   29
3    536365    84029G  KNITTED UNION FLAG HOT WATER BOTTLE  ...  2016     11   29
4    536365    84029E       RED WOOLLY HOTTIE WHITE HEART.  ...  2016     11   29

[5 rows x 10 columns]
A new column named Year,Month and Day is created.
```

CODE:

```
#%%
print("Adding a new variable TotalExpense to the dataset")
df['TotalExpense'] = df['Quantity'] * df['UnitPrice']
```

Result:

```
Adding a new variable TotalExpense to the dataset

  InvoiceNo StockCode  ... Day  TotalExpense
0    536365    85123A  ...  29         15.30
1    536365     71053  ...  29         20.34
2    536365    84406B  ...  29         22.00
3    536365    84029G  ...  29         20.34
4    536365    84029E  ...  29         20.34

[5 rows x 11 columns]
A new column named TotalExpense is created and added to the dataset.
```

III. Checking the Cleaned dataset

CODE:

```
#%%
print("Rechecking for null values:")
print(df.isnull().sum())
print('#',50*"-")
```

Result:

```
Dropping rows with missing/na values

Rechecking for null values:

InvoiceNo       0
StockCode       0
Description     0
Quantity        0
UnitPrice       0
CustomerID      0
Country         0
Year            0
Month           0
Day             0
TotalExpense    0
dtype: int64
```

We can see that all values are turned to zero meaning that there are no null values in the dataset.

Result:

```
Number of data points in the final cleaned dataset: 401472
# -----------------------------------------------------
```

III. Trial GUI (not included for the main group project)

A simple gui trial to save the cleaned dataset automatically to a preferred location can be useful.

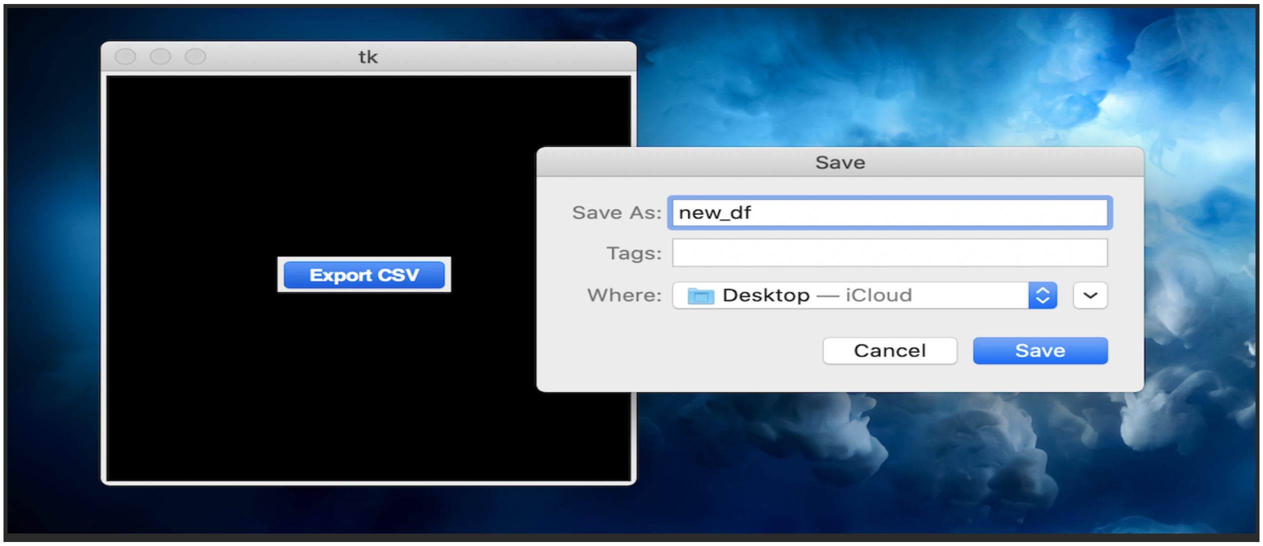
CODE:

```
#%% basic GUI Trial, cite:datafish
'''Here I am just exploring and adding a gui where the clean dataset created can be saved through an interactive session\n.
Here the user can just save the data in csv format after datacleaning'''
import tkinter as tk
from tkinter import filedialog
from pandas import DataFrame
df = DataFrame(df)

root = tk.Tk()
GUI = tk.Canvas(root, width=300, height=300, bg='black', relief='raised')
GUI.pack()

#exportCSV() function is used to convert existing to a new csv file
def exportCSV():
    global df
    export_file_path = filedialog.asksaveasfilename(defaultextension='.csv')
    df.to_csv(export_file_path, index=False, header=True)
#button
Button_CSV = tk.Button(text='Export CSV', command=exportCSV, bg='blue', fg='black',
                             font=('helvetica', 12, 'bold'))
GUI.create_window(150, 150, window=Button_CSV)
root.mainloop()
```

```
Result:
```



5. Summary and conclusion

Data cleaning is a crucial yet a long task of data mining. This process has helped me learn the importance of data cleaning how efficient the dataset turns out to be for modelling if the dataset is not noisy. Data cleaning and data preprocessing helps understand the problem statement, what insights need to be gained for building our model. Understanding the problem statement, having a domain knowledge on the dataset that will be worked on is important. When dealing with a huge dataset the complexity of data cleaning increases. Dataset is obtained from the real world is much more raw/unstructured data (e.g. Twitter comments). Larger datasets which is more noisy can be used for enhancing the data cleaning.

6. Calculation

Used from exact internet 15(cited), modified 10 (based on requirements of cleaning), added 130 (python codes taught in class). Cal: 3.448

7. References

Websites used:

https://pandas.pydata.org/

https://scikit-learn.org/stable/modules/generated/sklearn.cluster.KMeans.html

https://www.analyticsvidhya.com/blog/2019/08/comprehensive-guide-k-means-clustering/

https://towardsdatascience.com/understanding-k-means-clustering-in-machine-learning-6a6e67336aa1

https://towardsdatascience.com/clustering-metrics-better-than-the-elbow-method-6926e1f723a6

https://www.kdnuggets.com/2019/11/customer-segmentation-using-k-means-clustering.html