Business Report

DSBA Data Mining Project – Part 2
Segmentation using K-Means Clustering

Contents

List of Figures

List of Tables

Problem Statement

Digital Ads data

The ads24x7 is a Digital Marketing company which has now got seed funding of $10 Million. They are expanding their wings in Marketing Analytics. They collected data from their Marketing Intelligence team and now wants you (their newly appointed data analyst) to segment type of ads based on the features provided. Use Clustering procedure to segment ads into homogeneous groups.

The following three features are commonly used in digital marketing.

CPM = (Total Campaign Spend / Number of Impressions) * 1,000

CPC = Total Cost (spend) / Number of Clicks

CTR = Total Measured Clicks / Total Measured Ad Impressions x 100

Perform the following steps in given order:

- Read the data and perform basic analysis such as printing a few rows (head and tail), info, data summary, null values duplicate values, etc.
- Treat missing values in CPC, CTR and CPM using the formula given
- Check if there are any outliers
- Do you think treating outliers is necessary for K-Means clustering? Based on your judgement decide whether to treat outliers and if yes, which method to employ. (As an analyst your judgement may be different from another analyst)
- Perform z-score scaling and discuss how it affects the speed of the algorithm
- Perform clustering and do the following:
    - Make Dendrogram using WARD and Euclidean distance
    - Make elbow plot (up to n=10) and identify optimum number of clusters
    - Print silhouette scores for up to 10 clusters and identify optimum number of clusters
    - Profile the ads based on optimum number of clusters using silhouette score and your domain understanding.

      [Hint: Group the data by clusters and take sum or mean to identify trends in clicks, spend, revenue, CPM, CTR, & CPC based on Device Type. Make bar plots.]
- Conclude the project by providing summary of your learnings

Read the data and perform basic analysis such as printing a few rows (head and tail), info, data summary, null values duplicate values, etc.

Let us start by reading the data and extracting basic information:

Table 1: Data Information

Columns	Number of Records		Datatype
Timestamp	23066	non-null	object
InventoryType	23066	non-null	object
Ad-Length	23066	non-null	int64
Ad-Width	23066	non-null	int64
Ad Size	23066	non-null	int64
Ad Type	23066	non-null	object
Platform	23066	non-null	object
Device Type	23066	non-null	object
Format	23066	non-null	object
Available_Impressions	23066	non-null	int64
Matched_Queries	23066	non-null	int64
Impressions	23066	non-null	int64
Clicks	23066	non-null	int64
Spend	23066	non-null	float64
Fee	23066	non-null	float64
Revenue	23066	non-null	float64
CTR	18330	non-null	float64
CPM	18330	non-null	float64
CPC	18330	non-null	float64

The data set contains of about 23K records with 19 variables (6 float64, 7 int64, 6 object).

Dataset Head:

Table 2: Data first 5 five rows of the dataset (shown here as columns to save space)

	0	1	2	3	4
Timestamp	2020-9-2-17	2020-9-2-10	2020-9-1-22	2020-9-3-20	2020-9-4-15
InventoryType	Format1	Format1	Format1	Format1	Format1
Ad - Length	300	300	300	300	300
Ad- Width	250	250	250	250	250
Ad Size	75000	75000	75000	75000	75000
Ad Type	Inter222	Inter227	Inter222	Inter228	Inter217
Platform	Video	App	Video	Video	Web
Device Type	Desktop	Mobile	Desktop	Mobile	Desktop
Format	Display	Video	Display	Video	Video
Available_Impressions	1806	1780	2727	2430	1218
Matched_Queries	325	285	356	497	242
Impressions	323	285	355	495	242
Clicks	1	1	1	1	1
Spend	0.0	0.0	0.0	0.0	0.0
Fee	0.35	0.35	0.35	0.35	0.35
Revenue	0.0	0.0	0.0	0.0	0.0
CTR	0.0031	0.0035	0.0028	0.002	0.0041
CPM	0.0	0.0	0.0	0.0	0.0
CPC	0.0	0.0	0.0	0.0	0.0

Table 3: Data summary stats (for continuous variables only)

	count	mean	std	min	25%	50%	75%	max
Ad - Length	23066.0	385.16	233.65	120.00	120.00	300.00	720.00	728.00
Ad- Width	23066.0	337.90	203.09	70.00	250.00	300.00	600.00	600.00
Ad Size	23066.0	96674.47	61538.33	33600.00	72000.00	72000.00	84000.00	216000.00
Available_Impressions	23066.0	2432043.67	4742887.76	1.00	33672.25	483771.00	2527711.75	27592861.00
Matched_Queries	23066.0	1295099.14	2512969.86	1.00	18282.50	258087.50	1180700.00	14702025.00
Impressions	23066.0	1241519.52	2429399.96	1.00	7990.50	225290.00	1112428.50	14194774.00
Clicks	23066.0	10678.52	17353.41	1.00	710.00	4425.00	12793.75	143049.00

Spend	23066.0	2706.63	4067.93	0.00	85.18	1425.12	3121.40	26931.87
Fee	23066.0	0.34	0.03	0.21	0.33	0.35	0.35	0.35
Revenue	23066.0	1924.25	3105.24	0.00	55.37	926.34	2091.34	21276.18
CTR	18330.0	0.07	0.08	0.00	0.00	0.08	0.13	1.00
CPM	18330.0	7.67	6.48	0.00	1.71	7.66	12.51	81.56
CPC	18330.0	0.35	0.34	0.00	0.09	0.16	0.57	7.26

Minimum value for several variables is 0. There are no negative values. The CTR, CPM, and CPC are derived fields and have missing values. Note that the range of the values for different variables are very different.

There are no duplicate values in the data.

There are missing values in the 3 variables. Their counts are given in Figure 1.

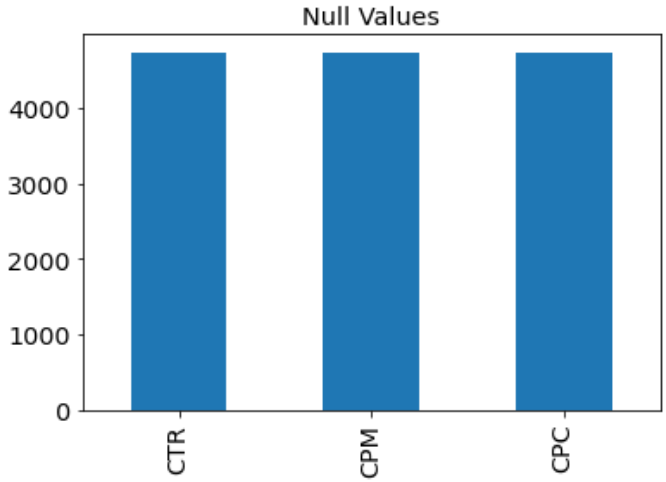


Figure 1: Missing Values Count representation using bar plot

Missing Value Treatment

Treat missing values in CPC, CTR and CPM using the formula given

CPM = (Total Campaign Spend / Number of Impressions) * 1,000

CPC = Total Cost / Number of Clicks

CTR = Total Measured Clicks / Total Measured Ad Impressions x 100

Check if there are any outliers. Do you think treating outliers is necessary for K-Means clustering? Based on your judgement decide whether to treat outliers and if yes, which method to employ. (As an analyst your judgement may be different from another analyst)

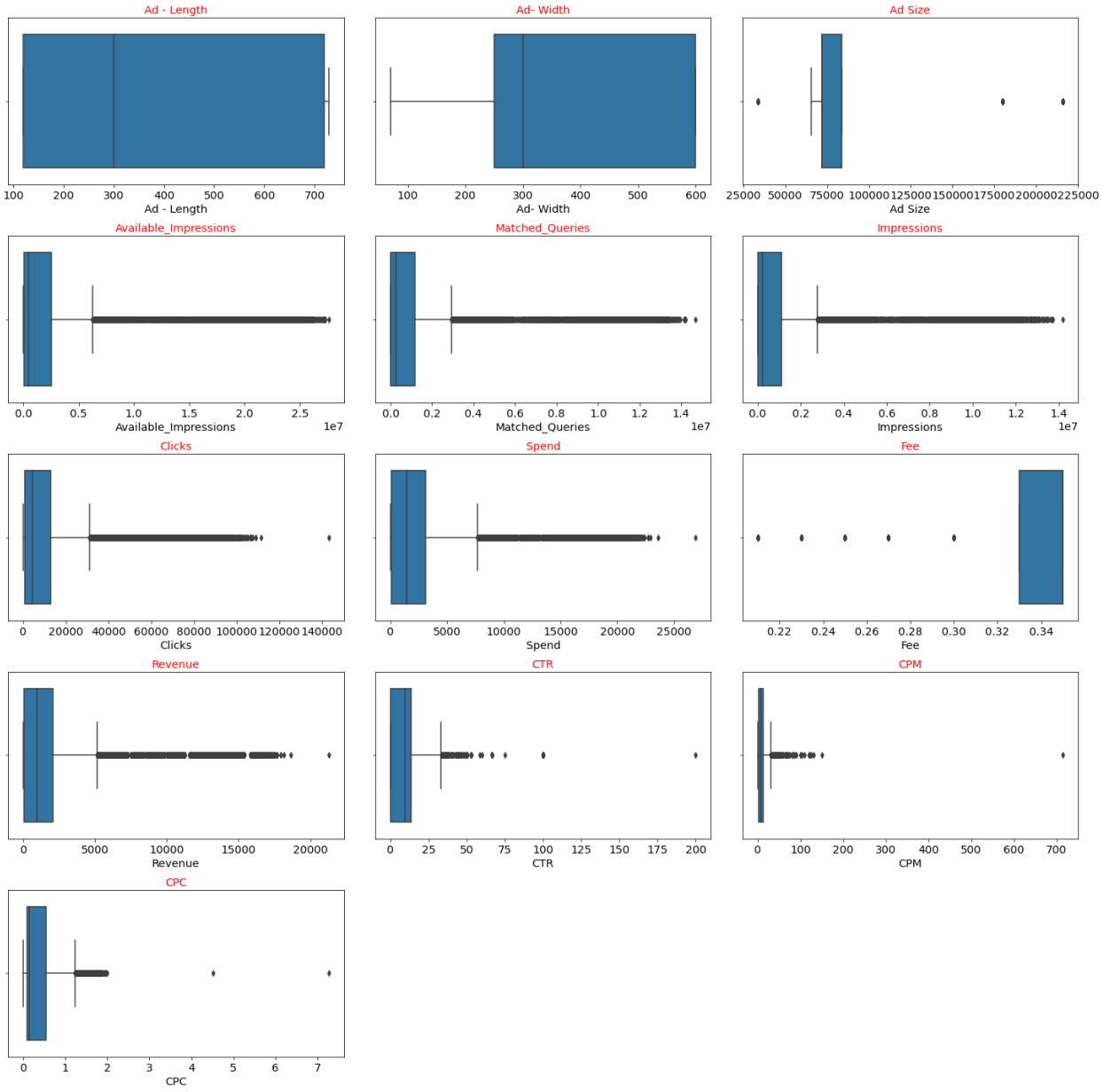


Figure 2: Boxplot for outliers

Although the k-means clustering technique is well-studied, it may have some limitations for real data. This is because the k-means objective assumes that all of the points can be naturally partitioned into k distinct clusters, which is often an unrealistic assumption in practice. Real data typically has contamination or

noise, and the k-means method may be sensitive to it. Noise can drastically change the quality of the clustering solution and it is important to take this into account in designing algorithms for partition [1].

Possible Approaches for reducing noise:

1. Treating outliers using IQR method.
2. Treating outliers using z-score method.
3. Using EDA results to segment data into two or more parts and then apply k-means algorithm to each part separately. For example: In Bank Customer Segmentation, High Net worth Individuals can be separated from Low Income Individuals and then separate models should be employed on all datasets. This method is applicable only if the size of the data is large and each part of the dataset has reasonable number of data points.

For this project, we will treat outliers using IQR Method, and compare results with model without outlier treatment.

Outlier Detection and Treatment using IQR method

In this method, any observation that is less than Q1 – 1.5 IQR or more than Q3 + 1.5 IQR is considered an outlier.

To treat outliers, we defined a function 'treat_outlier' where

- The larger values (>upper whisker) are all equated to the 95th percentile value of the distribution
- The smaller values (<lower whisker) are all equated to the 5th percentile value of the distribution.

Table 4: Summary Table for Outlier Detection and Treatment

Column Name	5th Percent	Q1	Q3	IQR	Q1 – 1.5IQR	Q3 + 1.5IQR	95th Percentile	Max	Min
Ad - Length	120.0	120.0	720.0	600.0	-780.0	1620.0	728.0	728.0	120.0
Ad- Width	70.0	250.0	600.0	350.0	-275.0	1125.0	600.0	600.0	70.0
Ad Size	33600.0	72000.0	84000.0	12000.0	54000.0	102000.0	216000.0	216000.0	33600.0
Available_Impressions	486.25	33672.25	2527711.75	2494039.5	-3707387.0	6268771.0	14363912.25	27592861	1.0
Matched_Queries	160.25	18282.5	1180700.0	1162417.5	-1725343.75	2924326.25	7803449.0	14702025	1.0
Impressions	149.25	7990.5	1112428.5	1104438.0	-1648666.5	2769085.5	7473380.25	14194774	1.0
Clicks	13.0	710.0	12793.75	12083.75	-17415.625	30919.375	50662.0	143049	1.0
Spend	1.03	85.18	3121.4	3036.22	-4469.15	7675.73	12899.76	26931.87	0.0
Fee	0.25	0.33	0.35	0.019	0.3	0.38	0.35	0.35	0.21
Revenue	0.6695	55.36	2091.34	2035.97	-2998.59	5145.29	9674.82	21276.18	0.0
CTR	0.184	0.265	13.47	13.205	-19.54	33.27	23.78	200	0.0108
CPM	1.194	1.75	13.04	11.29	-15.19	29.98	20.37	715	0.0
CPC	0.057	0.089	0.546	0.456	-0.595	1.23	0.925	7.264	0.0

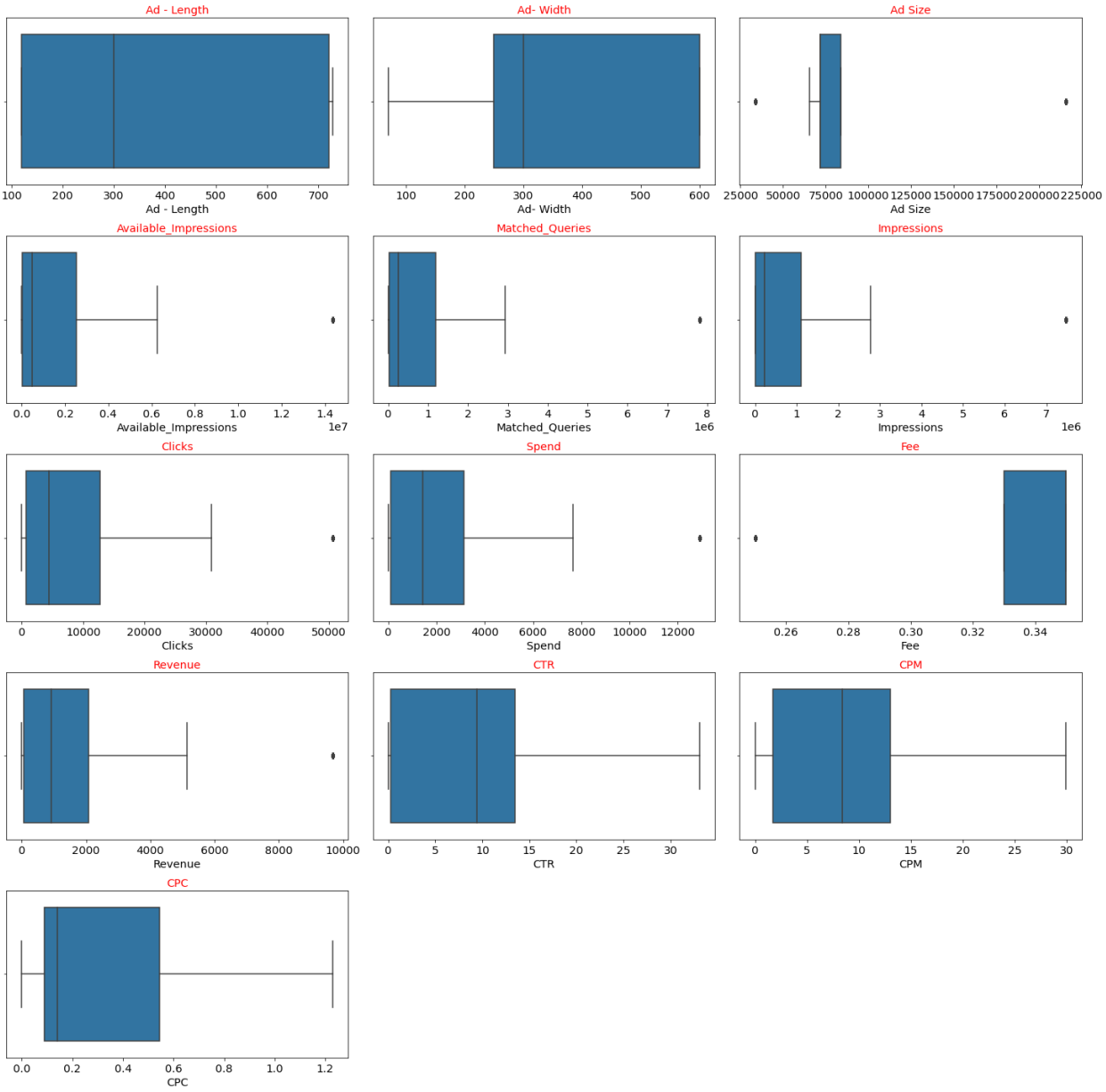


Figure 3: Boxplots after Outlier Treatment

Perform z-score scaling and discuss how does it affects the performance of the algorithm

We used scikit-learn's StandardScaler to perform z-score scaling. Table 6 shows the first five rows of the scaled data (rows transposed as columns)

Table 5: Scaled data (top 5 rows shown)

	0	1	2	3	4
Ad - Length	-0.364496	-0.364496	-0.364496	-0.364496	-0.364496
Ad- Width	-0.432797	-0.432797	-0.432797	-0.432797	-0.432797
Ad Size	-0.359227	-0.359227	-0.359227	-0.359227	-0.359227
Available_Impressions	-0.569484	-0.569490	-0.569269	-0.569339	-0.569622
Matched_Queries	-0.567061	-0.567076	-0.567049	-0.566994	-0.567093
Impressions	-0.563943	-0.563958	-0.563931	-0.563875	-0.563975
Clicks	-0.719779	-0.719779	-0.719779	-0.719779	-0.719779
Spend	-0.722776	-0.722776	-0.722776	-0.722776	-0.722776
Fee	0.487214	0.487214	0.487214	0.487214	0.487214
Revenue	-0.676118	-0.676118	-0.676118	-0.676118	-0.676118
CTR	-0.978830	-0.973650	-0.982332	-0.992329	-0.965826
CPM	-1.220346	-1.220346	-1.220346	-1.220346	-1.220346
CPC	-1.083011	-1.083011	-1.083011	-1.083011	-1.083011

Scaling of variables is important for clustering to stabilize the weights of the different variables. If there is wide discrepancy in the range of variables (refer to Table 3) cluster formation may be affected by weight differential.

The features contained in a data set may have different units (e.g. feet, kilometers, and hours) that, in turn, may mean that the variables have different scales. All machine learning algorithms are dependent on the scaling of data. If there is wide discrepancy among the input values, the unscaled model may be unstable, meaning that it may suffer from poor performance during learning and sensitivity to input values resulting in higher generalization error. [2]

One of the most common forms of pre-processing consists of a simple linear rescaling of the input variables.

— Page 298, Neural Networks for Pattern Recognition, 1995.

[2] https://machinelearningmastery.com/

Perform clustering

Perform Hierarchical by constructing a Dendrogram using WARD and Euclidean distance

Using SciPy's cluster hierarchy function, we created the below dendrogram.

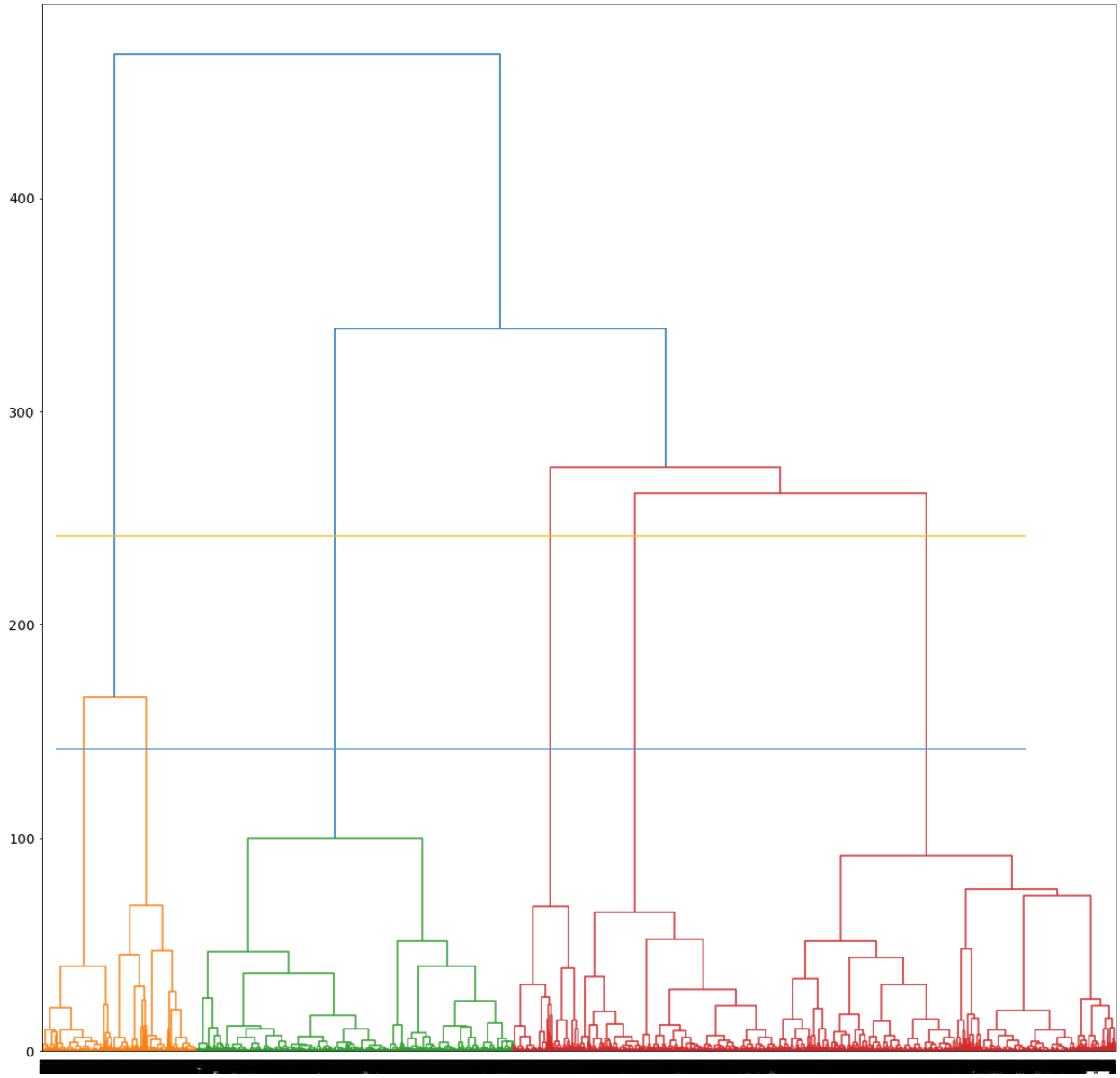


Figure 4: Dendrogram using WARD and Euclidean distance

In a Dendrogram, each branch is called a clade. The terminal end of each clade is called a leaf. The arrangement of the clades tells us which leaves are most similar to each other. The height of the branching points indicates how similar or different they are from each other: **the greater the height, the greater the difference.**

[reference - https://wheatoncollege.edu/wp-content/uploads/2012/08/How-to-Read-a-Dendrogram-Web-Ready.pdf]

Keeping the above reference as base, we can see the longest branch (tallest branch) is in blue. If we see that only blue, it will result in only 2 clusters which is not acceptable in business. If however the segmentation is at the tallest red branches, separated by the yellow horizontal line, 5 clusters are identified. Alternatively, there may be 3 clusters as well, designated by the yellow horizontal line. But we choose 5 Clusters using Dendrogram for this project.

Make Elbow plot (up to n=10) and identify optimum number of clusters for k-means algorithm

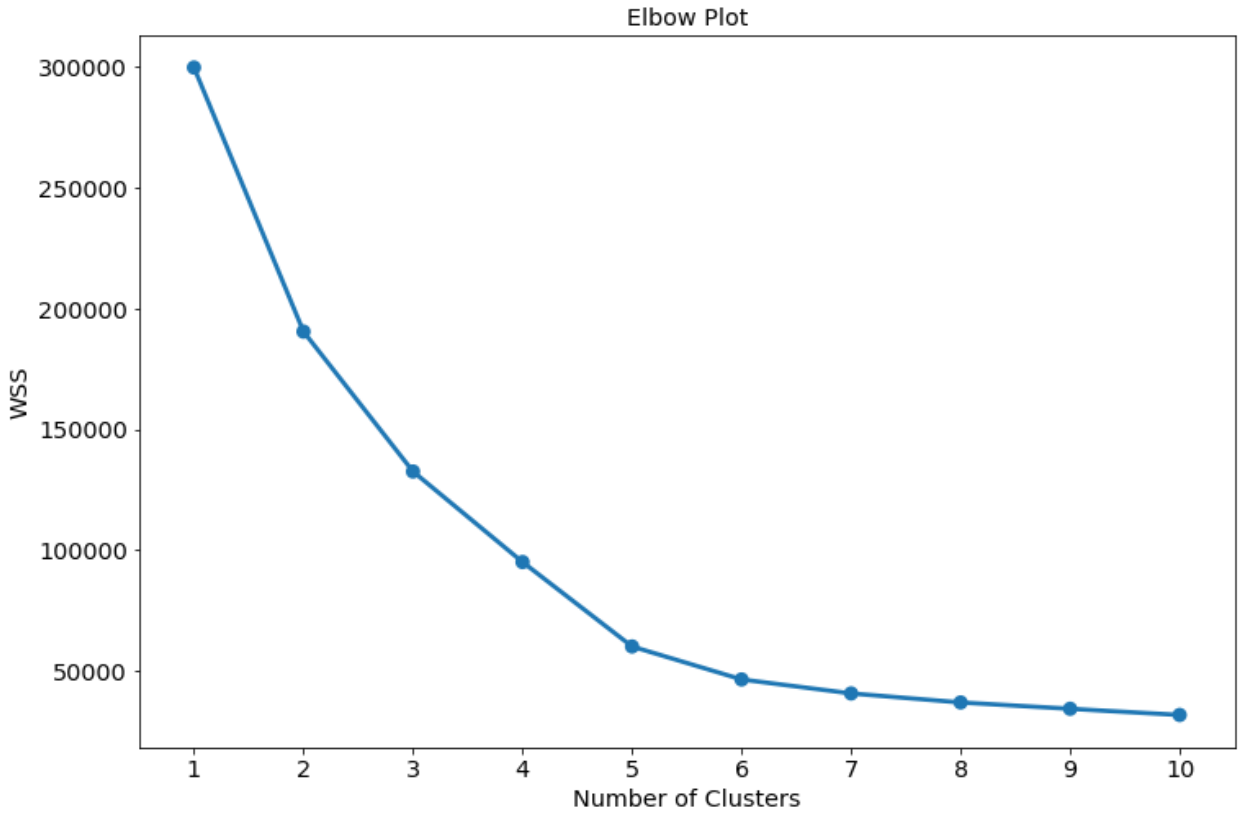


Figure 5: Elbow Plot

The optimum number of clusters is 5

Print silhouette scores for up to 10 clusters and identify optimum number of clusters

```
The Average Silhouette Score for 2 clusters is 0.47368
The Average Silhouette Score for 3 clusters is 0.39136
The Average Silhouette Score for 4 clusters is 0.45212
The Average Silhouette Score for 5 clusters is 0.55659
The Average Silhouette Score for 6 clusters is 0.57526
The Average Silhouette Score for 7 clusters is 0.53537
The Average Silhouette Score for 8 clusters is 0.46291
```

```
The Average Silhouette Score for 9 clusters is 0.45851
The Average Silhouette Score for 10 clusters is 0.46434
```

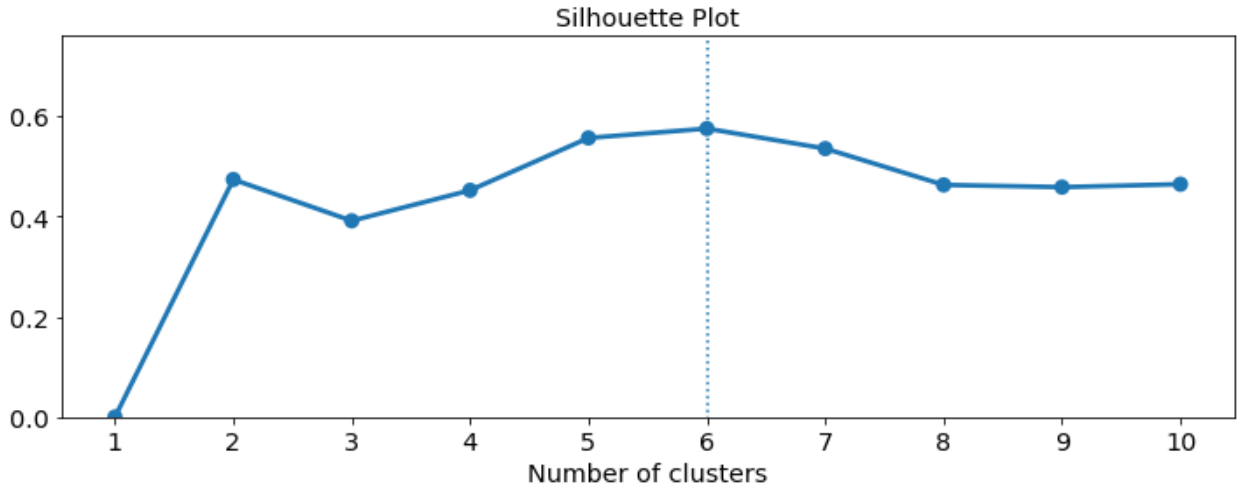


Figure 6: Silhouette Score Plot

Hierarchical Clustering as well as KMeans Clustering were ferformed. We used Elbow plot and Silhouette Score to identify optimum number of clusters in KMeans whereas in Hierarchical Clustering dendrogram was drawn. In Hierarchical method, we got 5 clusters while in KMeans, we got 5 (using elbow plot) and 6 clusters (using silhouette score).

Discussion (Non-graded)

We can always try alternative approaches to clustering using other linkage types and distance metrics for an exhaustive study of the data. Please refer to the Monograph for details. We observe that the methods used in this project yielded similar results i.e. with 5 clusters. (n_clusters=5 is also close with silhouette score of 0.518). According to the dendrogram in Figure 4, 3 or 4 clusters may also be considered. But more than 5 clusters may result in a high degree of fragmentation, where more than one clusters may have similar profiles. As per K-Means Silhouette score, we got 6 clusters. Hierarchical clustering may also indicate 6 clusters (see the blue horizontal line in Figure 4). There may be other possibilities also. However, the main considerations are:

1. What is the optimal number of clusters that support your business assumptions or rules about the market?
2. What is optimum number of market segments that may be handled in day-to-day operations?

As suggested in Rubric, we will segment the data into 6 clusters as per above plot (figure 6).

Profile the ads based on optimum number of clusters using silhouette score and your domain understanding

Table 6:Proportion of records per label

Label	Proportion
0	29.66
1	7.61
2	19.56
3	6.21
4	30.32
5	6.62

Table 7: : Cluster Profiles: Averages of the features considered

KMEANS_LABELS	0	1	2	3	4	5
Ad - Length	149.55	316.28	695.17	142.18	418.07	680.94
Ad- Width	558.21	254.54	316.80	571.18	157.14	117.92
Ad Size	75690.15	78364.78	213586.18	75625.96	56445.35	70159.76
Available_Impressions	46582.25	6583616.27	279059.43	843405.75	2070385.26	17858169.00
Matched_Queries	28661.60	3680737.02	147665.19	591156.63	1020575.04	9536142.81
Impressions	21257.39	3600777.01	126758.60	498760.11	980987.72	9181756.42
Clicks	2947.20	8548.28	13904.89	68157.27	3451.11	17394.94
Spend	318.92	4867.49	1224.16	7234.73	1763.33	15373.73
Fee	0.35	0.32	0.35	0.29	0.35	0.24
Revenue	208.48	3326.64	797.23	5205.55	1157.98	11761.38
CTR	15.97	0.24	13.64	13.77	0.39	0.19
CPM	14.71	1.38	11.92	15.13	1.80	1.71
CPC	0.10	0.60	0.09	0.11	0.58	0.92
freq	6842.00	1756.00	4514.00	1433.00	6994.00	1527.00

Observations:

1. The clusters 2 and 5 contain ads that have higher mean length than other clusters.
2. The clusters 0 and 3 have ads whose mean width is considerably more than the other clsuters
3. Cluster 5 has minimum ad size
4. Available impressions is highest for cluster 1
5. There is not much difference in Fee, but cluster 5 has very high mean spend and mean revenue compared to the others

(Many other observations can be added)

Discussion (non-graded):

It is possible to start with a larger number of clusters and based on comparison of the profiles, the number of clusters may be reduced. Note that the opposite is not possible. Cluster Plots are investigated to decide whether two clusters have considerable overlap, and therefore can be combined. Refer to the Monograph for details. In this case, Using KMeans, we can take 5 or 6 clusters, but what if we take 'n_clusters' upto 20 and we get better Silhouette score at, say, 14 clusters or 20 clusters. From a practical point of view, so many clusters are not usable. Hence, We take the best Silhouette score among a reasonable number of clusters, say 10, and compare cluster profiles and plots to trim the number of clusters. This is subjective and depends upon the domain application and experience of the user.

Often visualization helps to identify differences in clusters.

[Hint: Group the data by clusters and take sum or mean to identify trends in clicks, spend, revenue, CPM, CTR, & CPC based on Device Type. Make bar plots.]

Using the hint provided in the rubric, we will plot the bar charts by grouping the data by Cluster Labels and taking sum or mean of Clicks, Spend, Revenue, CTR,CPC, & CPM.

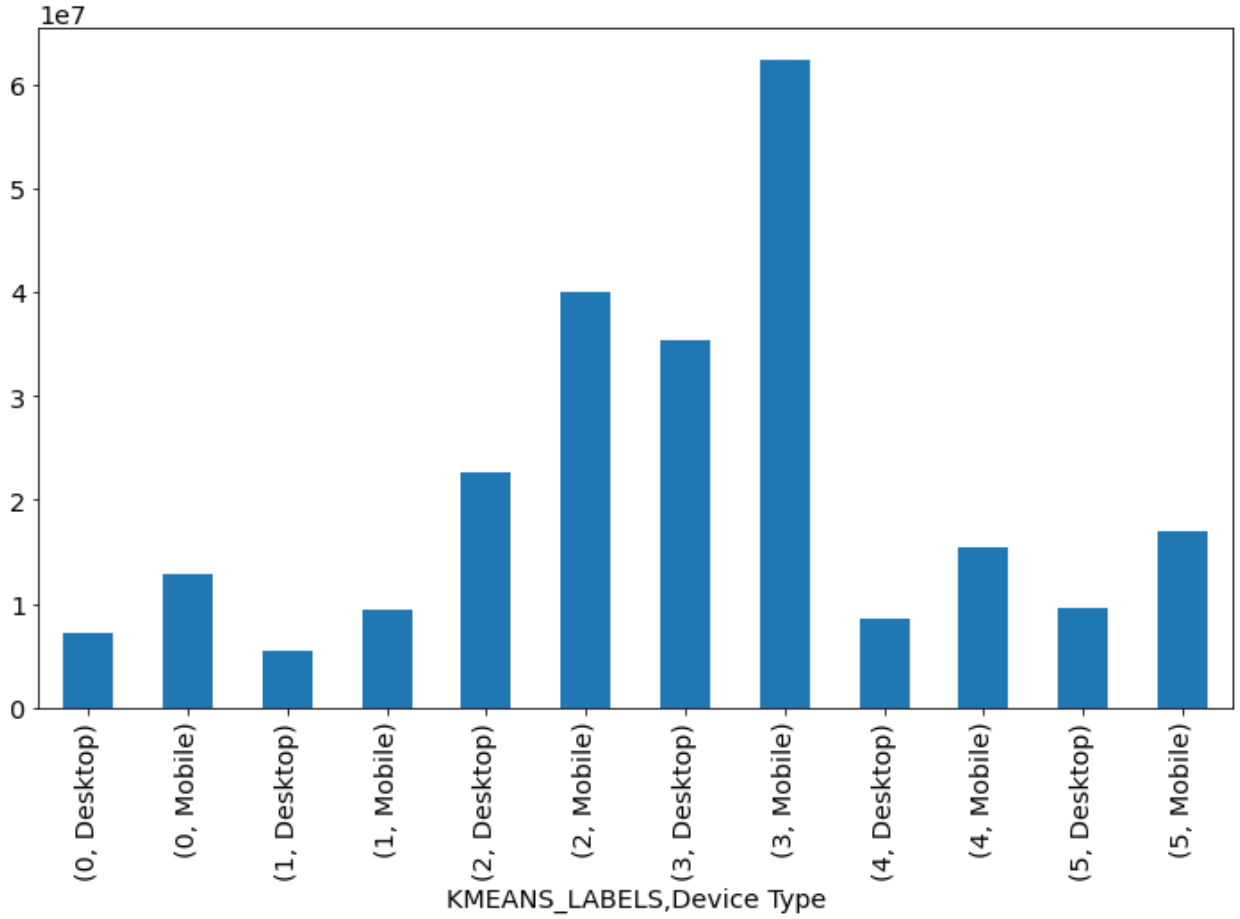


Figure 7: Comparison of Clusters according to device type (x-axis) and total clicks (y-axis)

Observation: The Mobile segment within Cluster 3 has the maximum number of clicks followed by Mobile segment within Cluster 2. Only for Cluster 3, desktop segment shows considerable number of clicks.

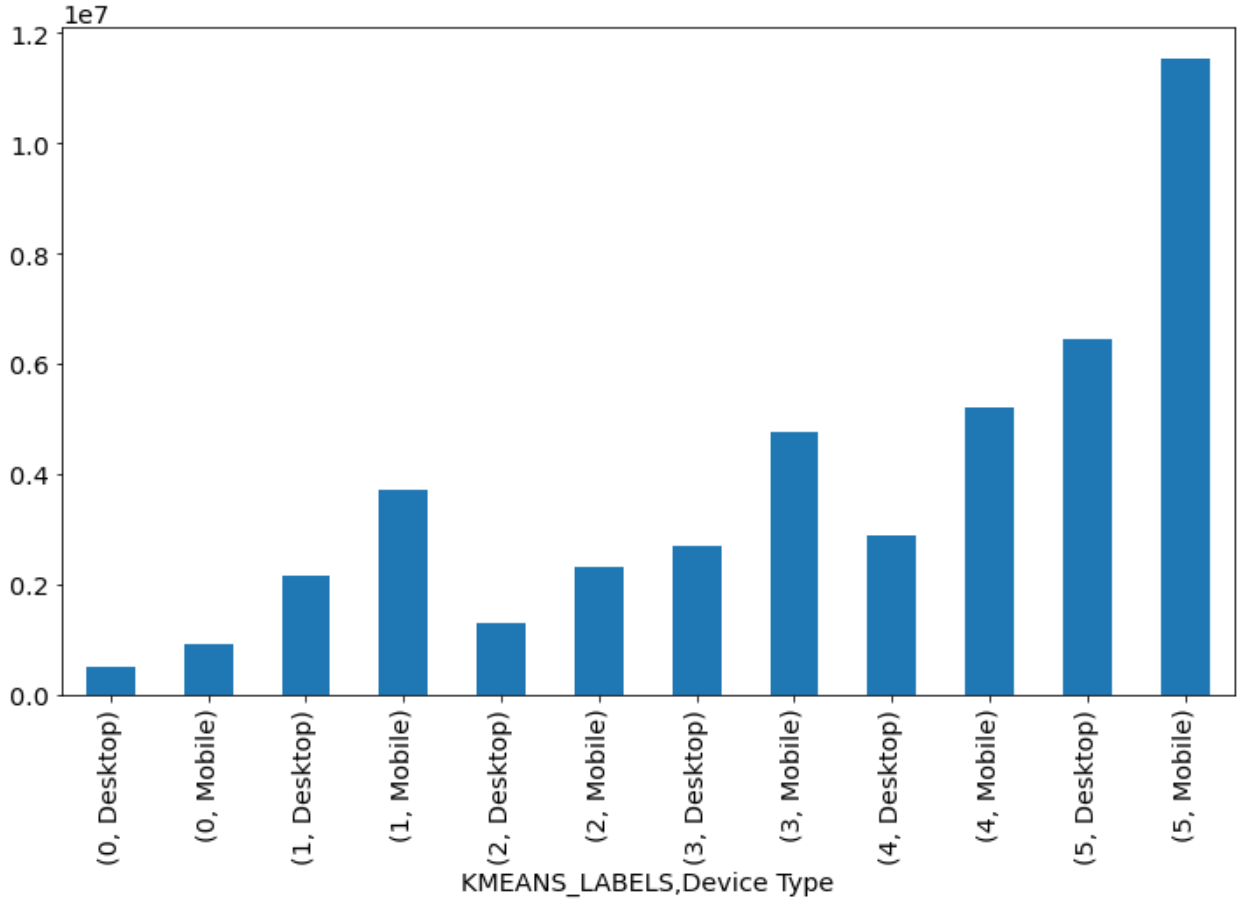


Figure 8: Comparison of Clusters according to device type (x-axis) and total revenue (y-axis)

Observations:

The mobile segment within Cluster 5 have most revenue generated and may be considered the best ads. Similarly, the desktop segment cluster label has highest revenue generated for Desktop Ads.

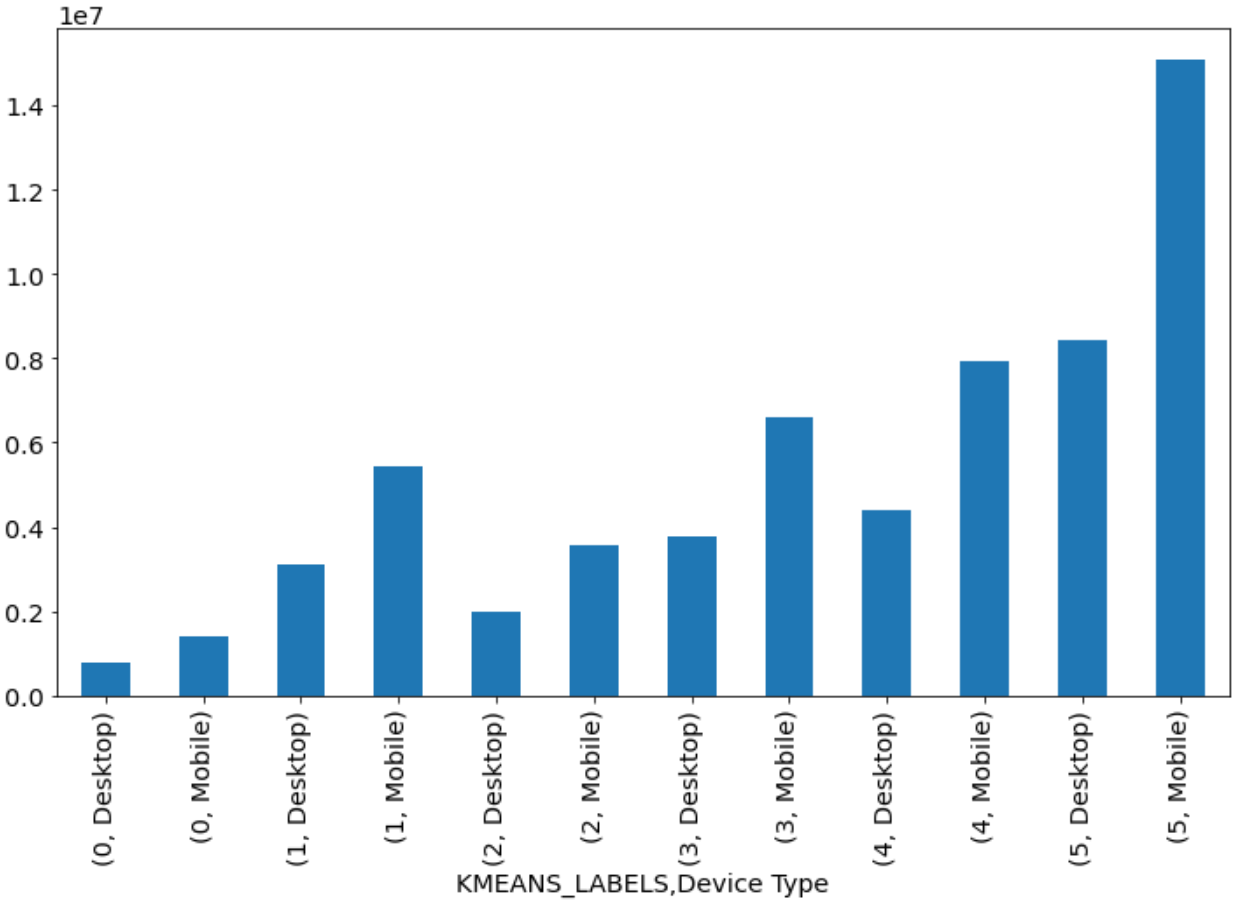


Figure 9: *Comparison of Clusters according to device type (x-axis) and* total spend (y-axis)*:*

Observations:

The mobile segment within cluster 5 show the highest total spending may be considered premium ads. Similarly, the desktop segment cluster label has highest spending done for Desktop Ads.

For Mobile segments clusters 3 and 4 show the most spending after cluster label 5 respectively.

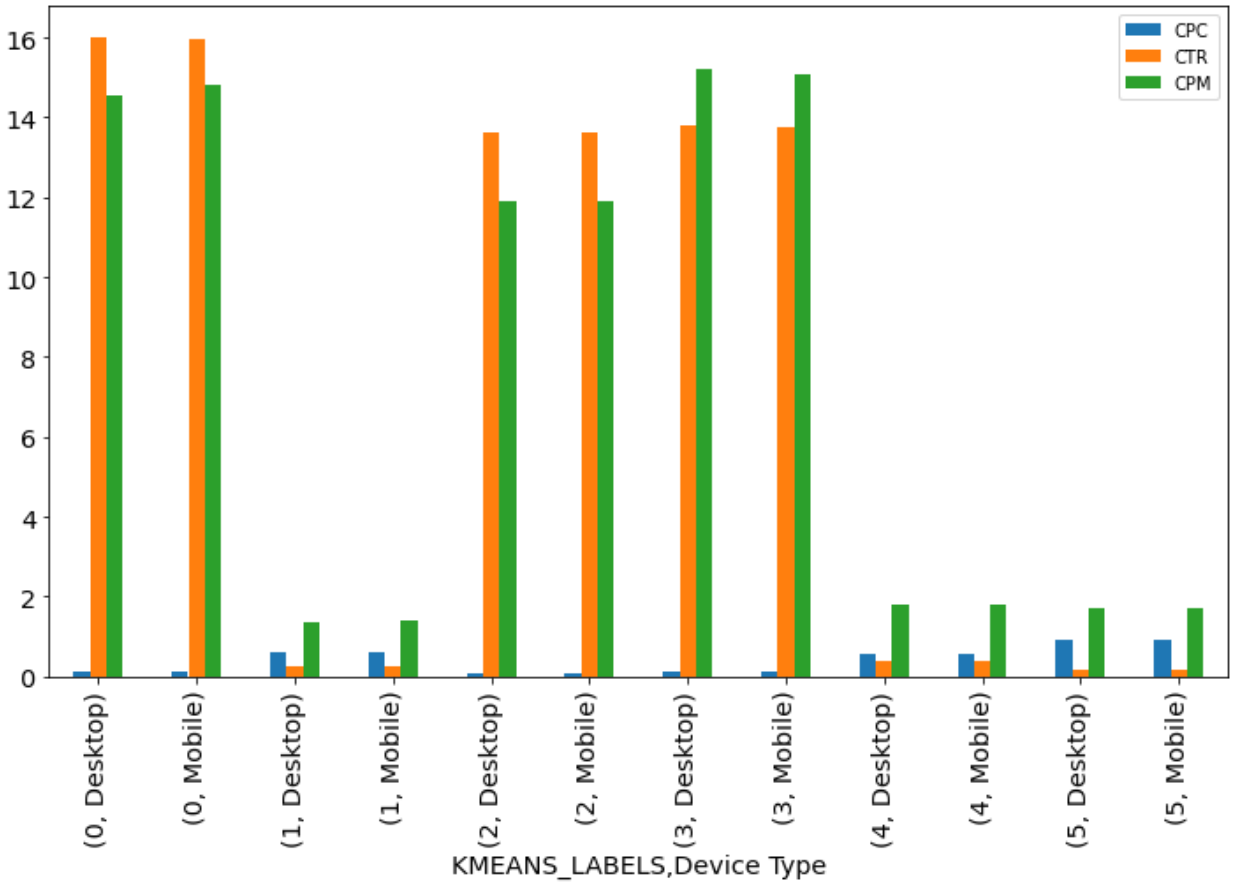


Figure 10: Comparison of Clusters according to device type (x-axis) and average CPC, CTR, CPM

CPM stands for "cost per 1000 impressions". In simple words, CPM refers to the amount it costs to have an ad published a thousand times on a website and is seen by users. For example, if a website publisher charges $4.00 CPM, that means an advertiser must pay $4.00 for every 1,000 impressions of its ads.

CPC stands for Cost Per Click. It is a method that websites use to determine the average times an advertiser has been clicked on the relevant ad. CPC is also a widely used google adwords metric that advertisers incorporate to manage their campaign budgets & performance. Let us say your CPC ads get 2 clicks, one costing $0.40 and the other is $0.20, this totals $0.60. You'd divide your $0.60 by 2 (your total number of clicks) to get an average CPC of $0.30.

CTR or Click Through Rate is measuring the success of online ads by aggregating the percentage of people that actually click on the ad to arrive at the hyperlinked website. For example, if an ad has been clicked 200 times after serving 50,000 times, by multiplying that result by 100 you get a click-through rate of 0.4%.

Reference: https://www.publift.com/adteach/what-are-cpm-cpc-cpa-ctr

Observations: According to Figure 10, Clusters 0, 2 and 3 have the highest avg CPM. These ads are probably posted on expensive and most visited websites. Average CTR is also the highest in the same

three clusters. There does not seem to be any considerable difference between the mobile and desktop segments here.

Selling ads according to **CPM puts a ceiling on revenue**. If you want to increase your revenue, you have to spend money on increasing your reach to create more ad opportunities, or pumping out more ads to the same users before seeing a return. But if you sell on **CTR**, **revenue is not capped**. You can increase engagement on the same number of impressions per person, or DAU (daily active user). Whereas with CPM, you stretch to reach more and more people, or degrade your user experience with more ads per user. Reference: https://blog.taboola.com/ctr-better-cpm-care/

Conclusion

In this project,

1. We learned to impute missing values using a different approach i.e. using custom formulae
2. We discussed about outlier's effect on quality of clustering profiles
3. We discussed about the scaling and its effect on performance of the algorithm
4. We discussed that clusters need to be revisited if there is too much similarity, or overlap, among them
5. We learned about certain digital marketing terms and their significance.

What more could be done?

- You can divide the data into Desktop and Mobile, then segment using clustering.
- You can dig deeper into clusters and generate more insight on which type of device or ad is more profitable in terms of spend and revenue, impressions, or clicks.
- You can learn more about Digital Marketing on Great Learning Academy for Free and utilize your Data Science Skills to help Digital Marketers utilize their data to the maximum.

Appendix

Code

In [1]:
```
import pandas as pd

import numpy as np

import seaborn as sns

from matplotlib import pyplot as plt
```

In [2]:
```
# Supress Warnings

import warnings
warnings.filterwarnings('ignore')
```

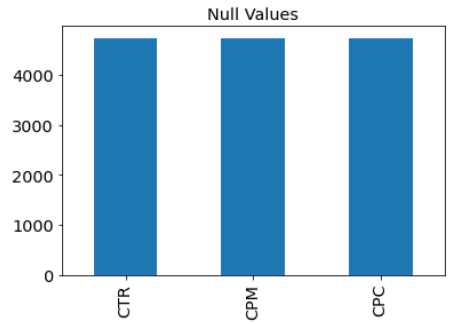
In [3]:
```
import matplotlib.pylab as pylab
params = {
         'axes.labelsize': 'x-large',
         'axes.titlesize':'x-large',
         'xtick.labelsize':'x-large',
         'ytick.labelsize':'x-large'}
pylab.rcParams.update(params)
```

In [4]:
```
df = pd.read_excel('C:/GL/DM/Data Mining New Projects/Data Mining New Projects/Clustering Clean Ads_Data.xlsx')
```

In [9]:
```
df.duplicated().sum()
```

Out[9]:
```
0
```

In [10]:
```
df.isnull().sum()[df.isnull().sum()>0].plot(kind='bar')
plt.title('Null Values');
```



missing values treatment

CPM = (Total Campaign Spend / Number of Impressions) * 1,000

CPC = Total Cost / Number of Clicks

CTR = Total Measured Clicks / Total Measured Ad Impressions x 100

[11]:
```
def calculate_ctr(x):
    clicks = df.Clicks
    impressions=df.Impressions
    ctr = (clicks/impressions)*100
    return ctr
```

[12]:
```
def calculate_cpc(x):
    spend=df.Spend
    clicks = df.Clicks
    cpc = (spend/clicks)
    return cpc
```

[13]:
```
def calculate_cpm(x):
    spend=df.Spend
    impressions=df.Impressions
    cpm = (spend/impressions)*1000
    return cpm
```

[14]:
```
df['CTR'] = df[['CTR']].apply(lambda x: calculate_ctr(x))
df['CPM']= df[['CPM']].apply(lambda x: calculate_cpm(x))
df['CPC']= df[['CPC']].apply(lambda x: calculate_cpc(x))
```

In [17]:
```
df['CTR'] = df[['CTR']].apply(lambda x: calculate_ctr(x))
df['CPM']= df[['CPM']].apply(lambda x: calculate_cpm(x))
df['CPC']= df[['CPC']].apply(lambda x: calculate_cpc(x))
```

In [18]:
```
df.isnull().sum()
```

Out[18]:
```
Timestamp               0
InventoryType           0
Ad - Length             0
Ad- Width               0
Ad Size                 0
Ad Type                 0
Platform                0
Device Type             0
Format                  0
Available_Impressions   0
Matched_Queries         0
Impressions             0
Clicks                  0
Spend                   0
Fee                     0
Revenue                 0
CTR                     0
CPM                     0
CPC                     0
dtype: int64
```

In [20]:
```
def treat_outlier(x):
    # taking 5,25,75 percentile of column
    q5= np.percentile(x,5)
    q25=np.percentile(x,25)
    q75=np.percentile(x,75)
    dt=np.percentile(x,95)
    #calculationg IQR range
    IQR=q75-q25
    #Calculating minimum threshold
    lower_bound=q25-(1.5*IQR)
    upper_bound=q75+(1.5*IQR)
    #Capping outliers
    return x.apply(lambda y: dt if y > upper_bound else y).apply(lambda y: q5 if y < lower_bound else y)
```

In [27]:
```
def print_outlier(x):
    # taking 5,25,75 percentile of column
    q5= np.percentile(x,5)
    q25=np.percentile(x,25)
    q75=np.percentile(x,75)
    dt=np.percentile(x,95)
    min_val = min(x)
    max_val = max(x)
    #calculationg IQR range
    IQR=q75-q25
    #Calculating minimum threshold
    lower_bound=q25-(1.5*IQR)
    upper_bound=q75+(1.5*IQR)
    #Capping outliers
    return ('5%=',q5,'Q1=',q25,'Q3=',q75,'IQR=',IQR,'LL=',lower_bound,'UL=', upper_bound, '95%', dt, 'max=',max_val, 'min=',min_
```

In [26]:
```
from sklearn.preprocessing import StandardScaler
```

In [27]:
```
X = StandardScaler()
```

In [28]:
```
scaled_df = pd.DataFrame(X.fit_transform(df_num),columns=df_num.columns)
```

In [29]:
```
scaled_df.head().T
```

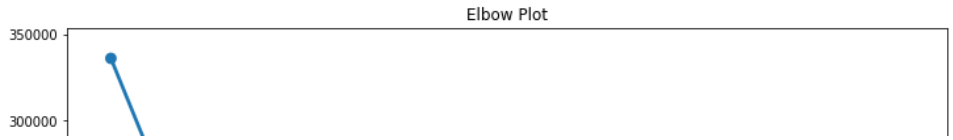
Out[29]:

	0	1	2	3	4
Ad - Length	-0.364496	-0.364496	-0.364496	-0.364496	-0.364496
Ad- Width	-0.432797	-0.432797	-0.432797	-0.432797	-0.432797
Ad Size	-0.359227	-0.359227	-0.359227	-0.359227	-0.359227
Available_Impressions	-0.569484	-0.569490	-0.569269	-0.569339	-0.569622
Matched_Queries	-0.567061	-0.567076	-0.567049	-0.566994	-0.567093
Impressions	-0.563943	-0.563958	-0.563931	-0.563875	-0.563975
Clicks	-0.719779	-0.719779	-0.719779	-0.719779	-0.719779
Spend	-0.722776	-0.722776	-0.722776	-0.722776	-0.722776
Fee	0.487214	0.487214	0.487214	0.487214	0.487214
Revenue	-0.676118	-0.676118	-0.676118	-0.676118	-0.676118
CTR	-0.978830	-0.973650	-0.982332	-0.992329	-0.965826

In [35]:
```
wss =[]
a=[1,2,3,4,5,6,7,8,9,10]
sil_score= []
for i in a:
    KM = KMeans(n_clusters=i,random_state=1)
    KM.fit(scaled_df)
    wss.append(KM.inertia_)
```

In [36]:
```
plt.figure(figsize=(12,8))
sns.pointplot(a, wss)
plt.title('Elbow Plot')
plt.xlabel('Number of Clusters')
plt.ylabel('WSS')
```

Out[36]: Text(0, 0.5, 'WSS')



In [35]:
```
ss={1:0}
for i in range(2, 11):
 clusterer = KMeans(n_clusters = i, init = 'k-means++', random_state = 1)
 y=clusterer.fit_predict(scaled_df)
 s =silhouette_score(scaled_df, y )
 ss[i]=round(s,5)
 print("The Average Silhouette Score for {} clusters is {}".format(i,round(s,5)))
```

```
The Average Silhouette Score for 2 clusters is 0.47368
The Average Silhouette Score for 3 clusters is 0.39136
The Average Silhouette Score for 4 clusters is 0.45212
The Average Silhouette Score for 5 clusters is 0.55659
The Average Silhouette Score for 6 clusters is 0.57526
The Average Silhouette Score for 7 clusters is 0.53537
The Average Silhouette Score for 8 clusters is 0.46291
The Average Silhouette Score for 9 clusters is 0.45851
The Average Silhouette Score for 10 clusters is 0.46434
```

```
maxkey= [key for key, value in ss.items() if value == max(ss.values())][0]
fig,ax = plt.subplots(figsize=(12,4))
sns.pointplot(list(ss.keys()),list(ss.values()))
plt.vlines(x=maxkey-1,ymax=0,ymin=0.75,linestyles='dotted')
ax.set(ylim=(0, 0.76))
ax.set_title('Silhouette Plot')
ax.set_xlabel('Number of clusters')
```

```
Text(0.5, 0, 'Number of clusters')
```

```
clusterer = KMeans(n_clusters =6, init = 'k-means++', random_state = 1)

clusterer.fit_predict(scaled_df)

labels = clusterer.labels_
```

```
df['KMEANS_LABELS'] = labels
```

```
df.head()
```

Cluster Profiling

In [40]:
```
df.KMEANS_LABELS.value_counts(1)*100
# Label has majority data
```

Out[40]:
```
4    30.321686
0    29.662707
2    19.569930
1     7.612937
5     6.620134
3     6.212607
Name: KMEANS_LABELS, dtype: float64
```

In [41]:
```
df.KMEANS_LABELS.value_counts()
```

Out[41]:
```
4    6994
0    6842
2    4514
1    1756
5    1527
3    1433
Name: KMEANS_LABELS, dtype: int64
```

```
clust_profile=df
clust_profile=clust_profile.groupby('KMEANS_LABELS').mean()
clust_profile['freq']=df.KMEANS_LABELS.value_counts().sort_index()
np.round(clust_profile,2).T
```

```
df.groupby(['KMEANS_LABELS','Device Type']).sum()['Clicks'].plot(kind='bar',figsize=(12,7));
```

```
df.groupby(['KMEANS_LABELS','Device Type']).sum()['Revenue'].plot(kind='bar',figsize=(12,7));
```

```
df.groupby(['KMEANS_LABELS','Device Type']).sum()['Spend'].plot(kind='bar',figsize=(12,7));
```

```
df.groupby(['KMEANS_LABELS','Device Type']).mean()[['CPC','CTR','CPM']].plot(kind='bar',figsize=(12,7));
```