Market Segmentation Project

Project by Aris Dressino, Master in Big Data & Management, October 2019

1a-b) Data Cleaning and Standardization

	duration	morning	afternoon	evening	age	weekend_day	female
1	212	0	1	0	17	0	0
2	229	0	1	0	17	0	0
3	259	0	0	1	17	1	1

We can notice that the dataframe is a multivariate/multidimensional set of 185k observations

```
mkt = pd.read_csv('MarketSegmentation.csv')

x = mkt.iloc[:].values

# # standardization mean = 0, sd=1 (1b)
from sklearn.preprocessing import StandardScaler
sc = StandardScaler()
x = sc.fit_transform(x)    # features' scaling

x.shape
```

```
(185190, 7)
```

Due to the dimensions and nature of the dataset, we need to **scale the variables' values by applying a standardization function.** (b) Done through Sklearn library's fit_transform function in order to redefine the **range of each single variable into a smaller interval (mean=0 and sd=1)** that will be easier to be computed for the subsequent machine learning algorithms. This will (a) **result in a training process well behaved because the numerical condition of the optimization problems is improved.** Moreover, it makes **training less sensitive to the scale of features, so we can better solve for coefficients** and obtain more accurate predictions. Theoretically, it is not always useful to standardize categorical variables, but the binary relation 0 and 1 of one-hot encoded variables is maintained. We use standardized distributions for categories in order to fit them better with other the other dimensions and avoid the scale to interfere with the model.

2a-b-c) Clustering with 10 Groups

Following the suggestions of my supervisor, I **apply a k-means algorithm in order to visualize 10 different clusters on the 7-dimensional space** we are studying. Using the **defined function count_clusters** (here on the left) We can observe (a) **10 groups with variable amount of observations** (104 in cluster 4, 9740 in 8 and 27681 in 9); I suppose that this is because **model is overfitting the data by creating a larger number of clusters than what is really needed**. To have a more precise idea, (b) we **extract the mean values of variables contained in the centroids** (following page)

```
# y_km

# function to count observations in the clusters
def count_clusters(ls):
    km = {}
    for n in ls:
        if (n in km):
            km[n] += 1
        else:
            km[n] = 1
    km = dict(sorted(km.items()))
    for key, value in km.items():
        print (" cluster % d : % d"%(key, value))
```

```
#### Fitting K-Means = 10 to the dataset

km10 = KMeans(n_clusters = 10, init = 'k-means++', random_state = 40)

y_km10 = km10.fit_predict(x)
```

```
y_km10
```

```
array([9, 9, 3, ..., 0, 0, 0], dtype=int32)
```

```
count_clusters(y_km10)    #
```

```
cluster  0 :  18282
cluster  1 :  15785
cluster  2 :  15822
cluster  3 :  7971
cluster  4 :  104
cluster  5 :  28495
cluster  6 :  44597
cluster  7 :  16713
cluster  8 :  9740
cluster  9 :  27681
```

```
# 10 centroids of 7 dimensions
km10.cluster_centers_
```

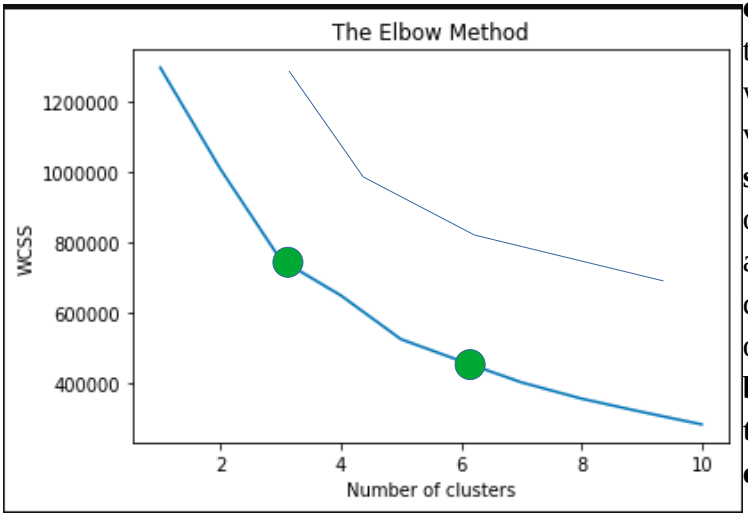
```
array([[-4.37037570e-02, -6.95701984e-01,  1.19771009e+00,
        -5.77071763e-01,  1.33416345e+00, -4.37597572e-01,
        -5.90125256e-01],
       [ 1.59496666e-02,  1.43739708e+00, -8.15991481e-01,
        -5.77071763e-01, -1.36924398e-02, -6.00334036e-02,
         1.67446738e+00],
       [ 2.95904290e-02, -6.95701984e-01,  1.14856289e+00,
        -5.77071763e-01, -2.99369916e-02, -4.52731568e-01,
         1.67446738e+00],
       [ 2.10268995e-02, -6.95701984e-01, -8.15991481e-01,
         1.73288673e+00, -3.76840561e-01,  2.20881439e+00,
         2.60729473e-02],
       [ 3.53492665e+01, -2.44469489e-01, -5.80434426e-01,
         7.11174320e-01, -4.96760016e-02, -1.76711718e-02,
        -7.44377049e-03],
       [-2.81014635e-02, -6.95701984e-01, -8.15991481e-01,
         1.73288673e+00, -2.13480546e-01, -4.52731568e-01,
        -5.97204825e-01],
       [-8.01562899e-02,  1.43677528e+00, -8.15991481e-01,
        -5.77071763e-01,  1.76084496e-01, -1.92109302e-01,
        -5.97204825e-01],
       [ 9.30630943e-02, -6.94680935e-01,  9.95982970e-01,
        -5.77071763e-01, -2.45231721e-01,  2.20881439e+00,
         9.26017007e-02],
       [ 5.78094296e-02, -6.95701984e-01, -8.15991481e-01,
         1.73288673e+00, -3.16755604e-01, -4.52731568e-01,
         1.67446738e+00],
       [-5.44504180e-02, -6.95701984e-01,  1.09533292e+00,
        -5.77071763e-01, -5.52338538e-01,  [illegible]-01,
        -5.97204825e-01]])
```

Screenshot

These are the normalized values of the centroids, we can **observe the presence of both positive and negative repeated values for the same attributes distributed across the clustering.** (c) It is interesting to observe the **cluster 4 contains a very small amount of observations: probably they are outliers using the scooter more than the average customer as we can see from the highest value associated to the duration of the rental** (underlined in the picture).

2d) Elbow Method and Appropriate Clustering

Minimizing the WCSS (within-cluster sums of squares) will maximize the distance between clusters. Said that, it is not easy do define the best number of groups from this graph: we can observe a constant decline in the value of the wcss, but **not a clear stabilization of the decreasing trend.** In other words, we should try to perform new analysis by choosing a different number of clusters for the k-means algorithm. In my opinion, **3 and 6 are important benchmarks and we should analyze them to discover if there is a more appropriate clustering size** for our market segmentation.



3a-b-c) K-means for less clusters?

The team will try to perform a k-means clustering for K = 6 and 3; an analysis of results will provide important insights in order to identify the relevant customers' groups for the research.

```
cluster  0 :  31484
cluster  1 :  53673
cluster  2 :  38236
cluster  3 :  45822
cluster  4 :  104
cluster  5 :  15871
```

```
# 6 centroids of 7 dimensions
km6.cluster_centers_
```

```
array([[ 6.64857932e-02, -2.41493620e-01,  1.52622559e-01,
         7.75478889e-03, -2.33527774e-01,  2.20881439e+00,
         9.48167785e-02],
       [-6.83510641e-02,  1.43739708e+00, -8.15991481e-01,
        -5.77071763e-01,  1.52461527e-01, -4.52731568e-01,
        -2.76890301e-02],
       [-5.76439790e-03, -6.95701984e-01, -8.15991481e-01,
         1.73282632e+00, -2.39795387e-01, -4.52731568e-01,
        -1.85332243e-02],
       [-5.13934085e-02, -6.95701984e-01,  1.13537285e+00,
        -5.77071763e-01,  1.88941457e-01, -4.52731568e-01,
        -5.97204825e-01],
       [ 3.53492665e+01, -2.44469489e-01, -5.80434426e-01,
         7.11174320e-01, -4.96760016e-02, -1.76711718e-02,
        -7.44377049e-03],
       [ 2.98910290e-02, -6.95701984e-01,  1.14845333e+00,
        -5.77071763e-01, -1.98079386e-02, -4.52731568e-01,
         1.67446738e+00]])
```

When k is forced to identify 6 clusters, we are able to (a) **observe a polarization of observations having a similar value across the groups.** Nevertheless, the cluster with **only 104 observations still remains: this denote an important feature of these outliers, looking to have characteristics very different from the other segments.** (b) The **centroids belong to the same scale and they maintain similar indicators** to the previous clusterization's algorithm. It is **possible to speculate about timing, durations of rentals (highest values in clusters 0,4,5), age and sex (females mainly in cluster 0) of customers** while searching for a model able to capture the right features of the users.

```
#### Fitting K-Means = 3 to the dataset <- appropriate
# with similar amount of observations

km3 = KMeans(
    n_clusters=3, init='k-means++',    # or 'random'
    n_init=12, max_iter=400,
    tol=1e-04, random_state=40)

y_km3 = km3.fit_predict(x)

count_clusters(y_km3)
```

```
cluster  0 :  60398
cluster  1 :  78511
cluster  2 :  46281
```

```
# 3 centroids of 7 dimensions
km3.cluster_centers_
```

```
array([[-4.38182900e-02,  1.43739708e+00, -8.15991481e-01,
        -5.77071763e-01,  1.26185253e-01, -1.57175611e-01,
        -3.35349676e-03],
       [ 6.90367326e-04, -6.95701984e-01,  1.10875107e+00,
        -5.77071763e-01,  5.82081025e-02,  1.17098468e-01,
         8.94286291e-03],
       [ 5.60129568e-02, -6.95655894e-01, -8.15991481e-01,
         1.73203823e+00, -2.63419400e-01,  6.47295303e-03,
        -1.07942485e-02]])
```

Differently, as we set the k to be equal to 3, (a) **the overfitting is now absent and each cluster has a comparable number of observations.** Outliers are absent and **cluster #1 has the most elements inside:** (b) **by decomposing its centroid, it is clear that these are women in their 20s that travel a lot mainly in the afternoon.** This is the average girl using the service and we should define a particular marketing campaign for her as she is our greatest source of revenues. **Cluster #0 represents younger men using the scooters in the morning during the week for a very short duration and spending less** (probably in order to commute to the working place) **and Cluster #1 is the users of the weekend.**

(c) Said that, it is clear that using 3 groups **is the best way to approach this segmentation problem. Not only it provides better information and insights regarding the users, but it allow to design different campaigns to address the nature of their habits.** It is really not bad!!