

Relatório

KMeans with Movie Lens 1 MB Dataset

<http://grouplens.org/datasets/movielens/>

The test was conducted using the small dataset which contains: 100,000 ratings and 1,300 tag applications applied to 9,066 movies by 671 users. Last updated 10/2016.

Data was massaged in Microsoft Excel before serving as input to the KMeans algorithm. The timestamp column was discarded. The columns used were: userId, movied and rating.

The data format was transformed as follows:

movied 1	user1 rating	user2 rating	user3 rating
movied 2	user1 rating	user2 rating	user3 rating
movied 3	user1 rating	user2 rating	user3 rating

Missing user ratings were filled by the average movie rating.

All the development was done using [Visual Studio Code](#) as the IDE and [Python](#) programming language with [scikit-learn](#) Machine Learning library and [NumPy](#)

Source code is available @ <https://github.com/leniel/DataMining>

Useful reading

Clustering <http://scikit-learn.org/stable/modules/clustering.html>

KMean <http://scikit-learn.org/stable/modules/generated/sklearn.cluster.KMeans.html>

PCA Decomposition

<http://scikit-learn.org/stable/modules/generated/sklearn.decomposition.PCA.html>

First test - [small sample of 100 movies' ratings] - NO PCA

This 1st test was conducted to better understand how the KMeans clustering algorithm works.

K = 15

The screenshot shows a Python IDE with a file named `kmeans.py` open. The code imports `KMeans` from `sklearn.cluster` and `genfromtxt` from `numpy`. It loads data from `\\Data\\ratings_massaged_100.csv` and fits a `KMeans` model with `n_clusters=15` and `random_state=0`. The results are printed: `kmeans.labels_`, `kmeans.cluster_centers_`, and `kmeans.inertia_`. The `DEBUG CONSOLE` shows the output of these operations. The `labels_` array is a 100x15 matrix of integers. The `cluster_centers_` array is a 15x100 matrix of floats. The `inertia_` value is 8747.72713856.

```
1 from sklearn.cluster import KMeans
2 from numpy import genfromtxt
3
4 import numpy as np
5
6 data = genfromtxt('\\\\Data\\ratings_massaged_100.csv', delimiter=',')
7
8 kmeans = KMeans(n_clusters=15, random_state=0).fit(data)
9
10 print(kmeans.labels_)
11
12 print(kmeans.cluster_centers_)
13
14 print(kmeans.inertia_)
```

DEBUG CONSOLE

```
[ 1 1 1 4 1 1 1 1 1 1 1 1 1 1 4 10 10 12 4 4 12 12 12 10
10 12 10 10 10 12 10 5 5 5 14 14 5 5 5 14 5 14 5 0 5 0 5 6 13
6 13 0 0 13 13 0 13 13 0 8 8 8 13 13 7 8 7 9 7 7 9 7 7 9
7 9 7 9 9 9 2 7 11 11 2 11 11 2 3 11 11 11 3 11 3 11 11 11 3]
[[ 54.83333333 3.03833333 3.03833333 ..., 3.03833333 3.03833333
3.03833333]
[ 7.41666667 3.52833333 3.57416667 ..., 3.52833333 3.53916667
3.6225 ]
[ 92. 1.63333333 1.63333333 ..., 1.63333333 1.63333333
1.63333333]
...,
[ 23.71428571 3.13714286 3.23428571 ..., 3.23428571 3.23428571
3.23428571]
[ 60.375 3.73625 3.57 ..., 3.73625 3.73625
3.73625 ]
[ 40.25 2.2575 2.2575 ..., 2.2575 2.2575
2.2575 ]]
8747.72713856
```

The labels property identifies the clusters, that is, each of the 100 movies is labeled with a given cluster number\identifier.

For K = 15, Inertia (inter-group distance) = 8747.72.

Inertia is not a normalized metric: we just know that lower values are better and zero is optimal. But in very high-dimensional spaces, Euclidean distances tend to become inflated (this is an instance of the so-called “curse of dimensionality”). Running a dimensionality reduction algorithm such as PCA prior to k-means clustering can alleviate this problem and speed up the computations.

K = 30

The screenshot shows a Python IDE with a file named `kmeans.py` open. The code imports `KMeans` from `sklearn.cluster` and `genfromtxt` from `numpy`. It loads data from `ratings_massaged_100.csv` and fits a `KMeans` model with `n_clusters=30` and `random_state=0`. The results are printed: `kmeans.labels_`, `kmeans.cluster_centers_`, and `kmeans.inertia_`. The `DEBUG CONSOLE` at the bottom displays the output of these print statements. The labels are a list of 100 integers representing cluster assignments. The cluster centers are a 30x7 matrix of floats. The inertia is a single float value, 4302.7037667.

```
1 from sklearn.cluster import KMeans
2 from numpy import genfromtxt
3
4 import numpy as np
5
6 data = genfromtxt('Data\\ratings_massaged_100.csv', delimiter=',')
7
8 kmeans = KMeans(n_clusters=30, random_state=0).fit(data)
9
10 print(kmeans.labels_)
11
12 print(kmeans.cluster_centers_)
13
14 print(kmeans.inertia_)
```

DEBUG CONSOLE

```
[19 14 14 16 14 22 14 22 14 1 22 14 22 1 20 22 22 1 20 20 1 1 13 13 21
21 13 21 21 21 13 21 15 15 3 4 4 15 3 3 4 15 28 15 28 3 28 12 25 7
24 7 7 18 12 12 18 7 12 18 10 10 2 5 2 10 2 5 2 2 5 27 27 0
27 9 27 9 9 8 27 29 29 26 29 23 17 11 23 26 29 11 26 11 6 6 6 11]
[[ 80.      4.63      4.63      ..., 4.63      4.63      4.63      ]
 [ 17.      3.418     3.528     ..., 3.418     3.418     3.418     ]
 [ 72.      3.21      3.21      ..., 3.21      3.21      3.21      ]
 ...,
 [ 81.8      3.06      3.06      ..., 3.06      3.06      3.06      ]
 [ 46.      2.93333333 2.93333333 ..., 2.93333333 2.93333333
 2.93333333]
 [ 93.75     3.495     3.495     ..., 3.495     3.495     3.495     ]]
4302.7037667
```

For K = 30, Inertia (inter-group distance) = 4302.70, that is, half the inertia of K = 15.

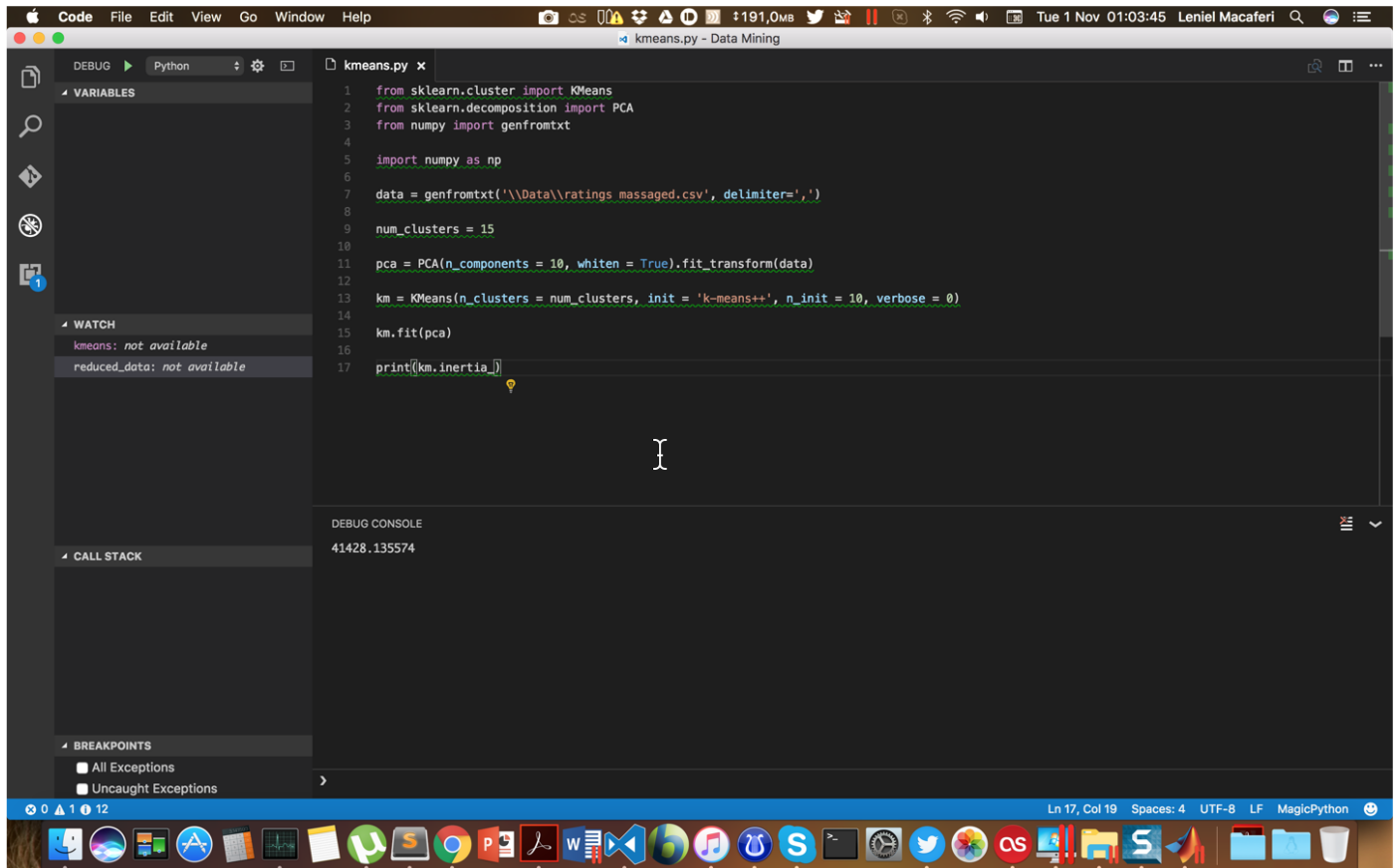
Second test - [full sample of 9066 movies' ratings] - NO PCA

For K = 15, Inertia = 35,906,782,235.4

For K = 30, Inertia = 7,642,479,407.69

Inertia was reduced by 5 with a double K.

Third test - [full sample of 9066 movies' ratings] - WITH PCA

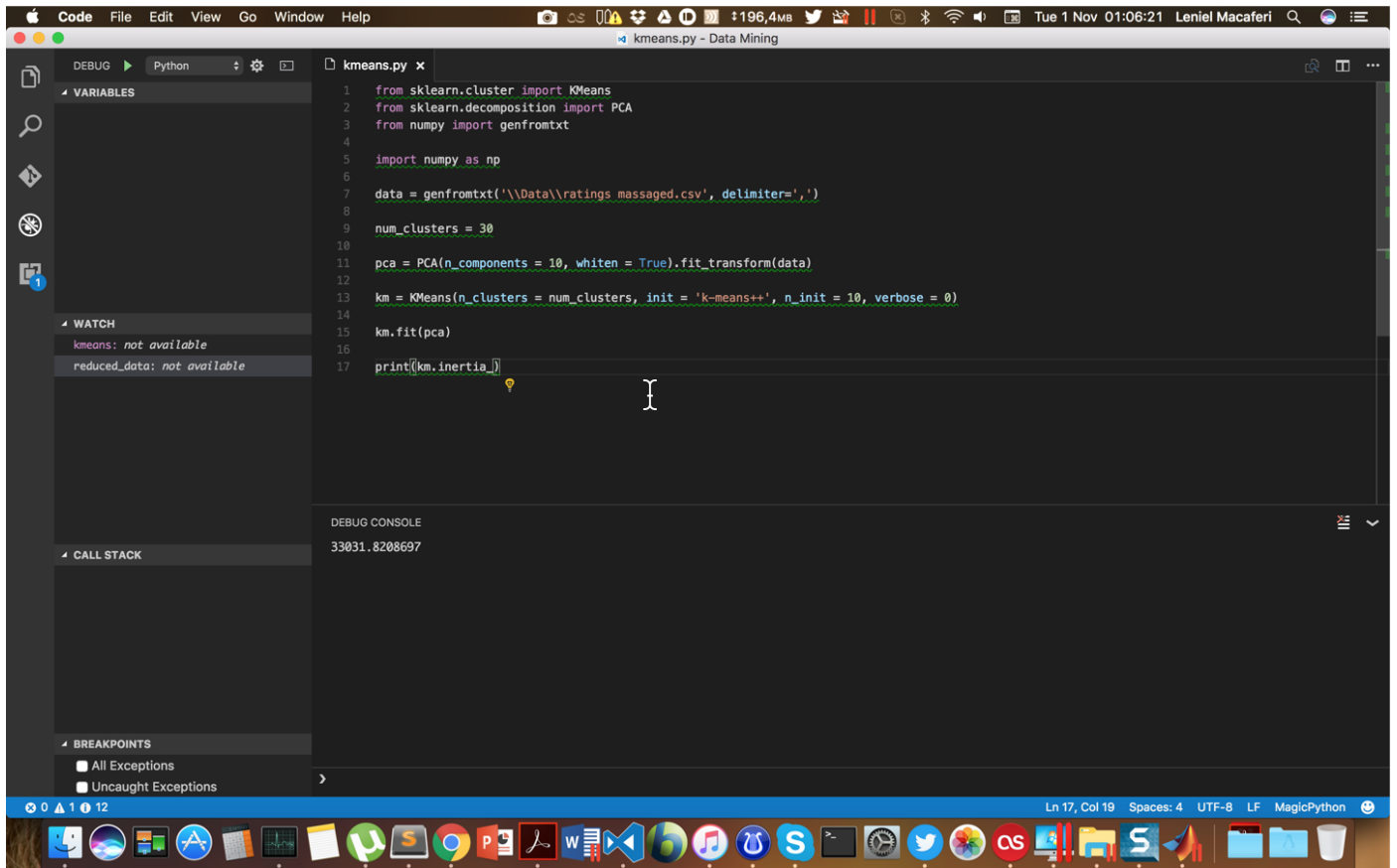


```
1 from sklearn.cluster import KMeans
2 from sklearn.decomposition import PCA
3 from numpy import genfromtxt
4
5 import numpy as np
6
7 data = genfromtxt('Data\\ratings massaged.csv', delimiter=',')
8
9 num_clusters = 15
10
11 pca = PCA(n_components = 10, whiten = True).fit_transform(data)
12
13 km = KMeans(n_clusters = num_clusters, init = 'k-means++', n_init = 10, verbose = 0)
14
15 km.fit(pca)
16
17 print(km.inertia_)
```

DEBUG CONSOLE

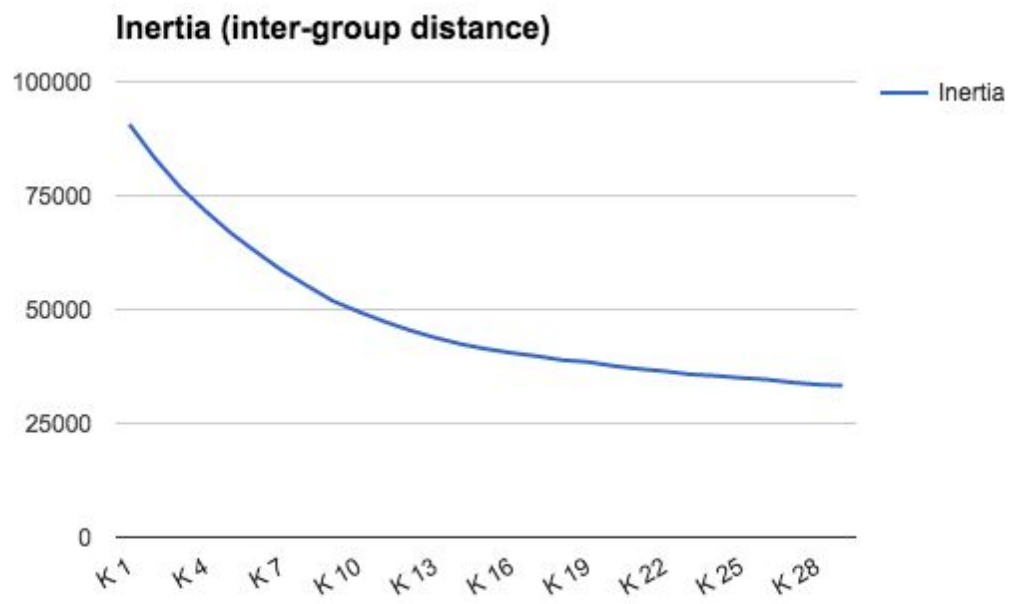
41428.135574

For PCA = 10 dimensions and K = 15, Inertia = 41428.13



For PCA = 10 dimensions and K = 30, Inertia = 33031.82

Graph showing how inertia decreases for greater K



Generated clusters are shown in the file clusters.txt.

Cluster 1 contains a good batch of animation movies.

Cluster 27 has 2 Star Wars movies: Episode V - The Empire Strikes Back (1980)' and Episode VI - Return of the Jedi (1983)