CAP 5768: Intro to Data Science, Fall 2023

Homework 2: Python and R

Due Sep 28 via Canvas by 6:00 PM. Submit file named: YourLastName-HW2.pdf

---

6. (15 points) Carefully study the Python notebook called MovieLens1M.ipynb. The data file required to run this notebook is in the ml-1m directory under data provided to you on Canvas. After you have understood the code in the notebook, modify the notebook to answer the following questions. You can do the work in Python or R. Submit the modified Python/R notebook.

   (a) Each movie in the database has a genre (comedy, animations, etc.) associated with it. Show the top 20 genres with the highest number of responses from users.

```python
group_genres = data.groupby('genres', observed=False).size().sort_values(ascending=False)
top20_genres=group_genres[:20].index
```

✓ 0.0s

Q1

a)

```python
top20_genres
```

✓ 0.0s

```
Index(['Comedy', 'Drama', 'Comedy|Romance', 'Comedy|Drama', 'Drama|Romance',
       'Action|Thriller', 'Horror', 'Drama|Thriller', 'Thriller',
       'Action|Adventure|Sci-Fi', 'Drama|War', 'Action|Sci-Fi',
       'Action|Sci-Fi|Thriller', 'Action', 'Action|Drama|War', 'Crime|Drama',
       'Comedy|Drama|Romance', 'Action|Adventure', 'Action|Drama',
       'Comedy|Horror'],
      dtype='object', name='genres')
```

   (b) Show the top 20 genres sorted by average ratings.

b)

```
top20_mean_ratings= mean_ratings3.loc[top20_genres
top20_mean_ratings
```

✓ 0.0s

	rating
genres	
Comedy	3.464456
Drama	3.780611
Comedy|Romance	3.530905
Comedy|Drama	3.720559
Drama|Romance	3.605417
Action|Thriller	3.525917
Horror	3.071932
Drama|Thriller	3.782552
Thriller	3.555879
Action|Adventure|Sci-Fi	3.381375
Drama|War	4.098936
Action|Sci-Fi	3.214201
Action|Sci-Fi|Thriller	3.664281
Action	3.354886
Action|Drama|War	4.047693
Crime|Drama	3.947094
Comedy|Drama|Romance	3.675129
Action|Adventure	3.676814
Action|Drama	3.561067
Comedy|Horror	3.357195

(c) Show the top 20 movies sorted by descending mean female ratings for a specific genre (say Drama).

```
mean_ratings4 = v.pivot_table('rating', index='title',columns='gender', aggfunc='mean').sort_values(by='F',ascending=False)
mean_ratings4
```

✓ 0.0s

gender	F	M
title		
I Am Cuba (Soy Cuba/Ya Kuba) (1964)	5.0	4.750000
Song of Freedom (1936)	5.0	NaN
Woman of Paris, A (1923)	5.0	2.428571
Ballad of Narayama, The (Narayama Bushiko) (1958)	5.0	3.428571
Gambler, The (A Játékos) (1997)	5.0	3.166667
...	...	...
War at Home, The (1996)	NaN	2.500000
Wend Kuuni (God's Gift) (1982)	NaN	4.000000
White Boys (1999)	NaN	1.000000
Windows (1980)	NaN	1.000000
Wooden Man's Bride, The (Wu Kui) (1994)	NaN	3.000000

759 rows × 2 columns

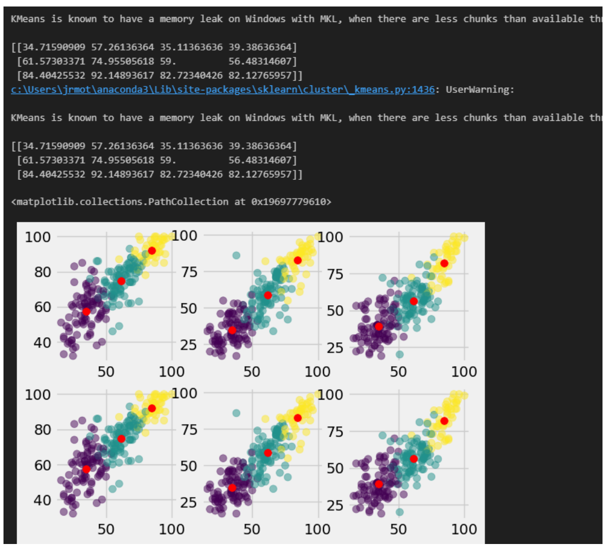
7. (**Extra Credit**) If you submitted a Python solution for the above problem, then rewrite the code in R and submit the R notebook. (Obviously, if you submitted an R notebook for the above problem, then rewrite and submit a Python notebook.) Make sure your program runs properly and gives the same answer as the Python program it emulates. Submit the R notebook as a separate file. The name of all files you submit should

include your name. For example, my files would be named GiriNarasimhan HW2.pdf and GiriNarasimhan HW2.Rmd

8. (10 points) In class, we discussed the MDCPS data set (see 5.3-MDCPS-Grades-2017.ipynb).

In class, we discussed clustering of the data set based on 4 features English Language Arts Achievement, Social Studies Achievement, Mathematics Achievement, Science Achievement (see 7.2-Clustering.ipynb). Study K-Means options from:

https://scikit-learn.org/stable/modules/generated/sklearn.cluster.KMeans. html. Redo the K-Means clustering by choosing (a) n init = 8, and (b) elkan algorithm instead of lloyd. Remember to set the seed to achieve reproducibility of your results.



I'n not sure if I was supposed to have the same clusters when changing the algorithm and n_init but the only difference I can see is in execution time.

9. (10 points) For the above problem, implement a reasonable quality measure to compare the different clusterings above. Display your results.

As mentioned before, the only difference the output gives is in execution time and shows that the Elkan algorithm is more efficient when finding the clusters. I tried silhouette score but it still gave the same average score.

Silhouette:

Default:

```
from sklearn.metrics import silhouette_score

kmeans = KMeans(n_clusters = 3, n_init = 10, random_state = 87, algorithm = 'lloyd')
y_means = kmeans.fit_predict(df)

silhuette_ave = silhouette_score(df, y_means)
print("Average Silhouette score: ", silhuette_ave)
✓ 0.0s
Average Silhouette score:  0.4062543255096628
```

Elkan and n_init=8: (even separated as elkan in one run and n_init=8 in another still showed same results)

```
from sklearn.metrics import silhouette_score

kmeans2 = KMeans(n_clusters = 3, n_init = 8, random_state = 87, algorithm = 'elkan')
y_means = kmeans2.fit_predict(df)

silhuette_ave = silhouette_score(df, y_means)
print("Average Silhouette score: ", silhuette_ave)
✓ 0.0s
Average Silhouette score:  0.4062543255096628
```

Timed comparison:

For Default:

```
252 ms ± 8.86 ms per loop (mean ± std. dev. of 7 runs, 1 loop each)
```

For elkan algorithm and n_init=8

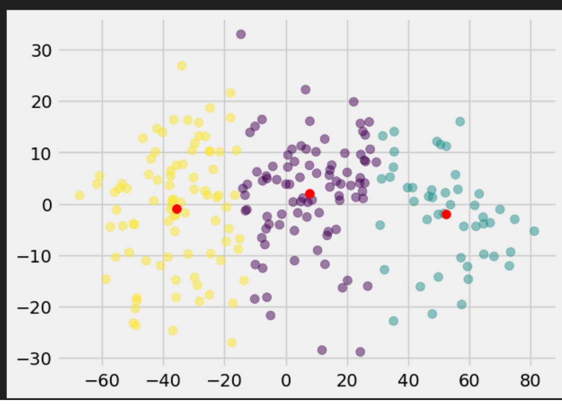
```
24.5 ms ± 475 µs per loop (mean ± std. dev. of 7 runs, 10 loops each)
```

10. (10 points) In class, we discussed 6.1-PCA.ipynb, where we ran PCA on the MDCPS data set. Follow it up by doing K-Means clustering on the points transformed by PCA (i.e., using pca 0 and pca 1). State and explain your parameter choices. Display your results by coloring points according to their cluster.
Parameters Chosen:
Components for PCA = 2 as permutation test showed only the first two components are relevant
Clusters = 3 as elbow method showed either 3 or 4 should be good enough clusters

Algorithm = 'LLoyd' as suggested in sklearn Elkan algorithm is efficient only with datasets with well-defined clusters and as seen on the first graph of PCA it looks like a cloud with not easy to locate clusters.

```
kmeans3 = KMeans(n_clusters = 3, n_init = 10, random_state = 87, algorithm = 'lloyd').fit(df_pca)
centroids3 = kmeans3.cluster_centers_
print(centroids3)

plt.scatter(df_pca['pca_0'], df_pca['pca_1'],
            c=kmeans3.labels_.astype(float), s=50, alpha=0.5)
plt.scatter(centroids3[:,0], centroids3[:, 1], c='red', s=50)
```

```
c:\Users\jrmot\anaconda3\Lib\site-packages\sklearn\cluster\_kmeans.py:1436: UserWarning:

KMeans is known to have a memory leak on Windows with MKL, when there are less chunks than available threa

[[  7.63121004   2.0206237 ]
 [ 52.45651166  -2.04700974]
 [-35.73447433  -0.95029604]]

<matplotlib.collections.PathCollection at 0x19696c45d50>
```



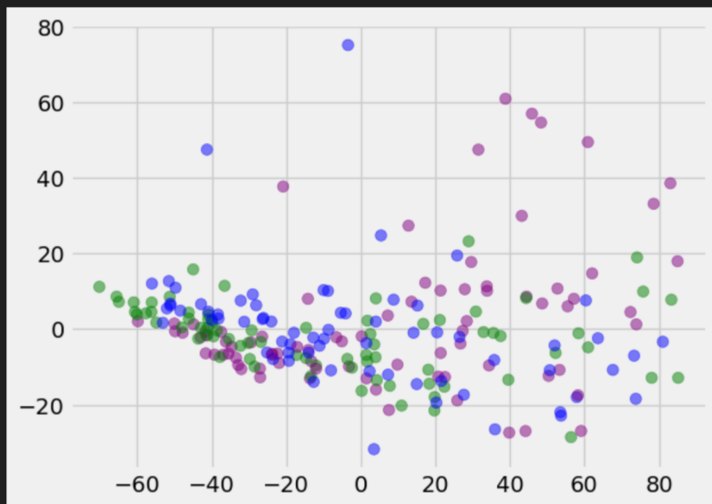
11. (5 points) Redo the PCA experiment with the MDCPS data, but color the points according to School Type. Try it again, but coloring the points according to Percent of Economically Disadvantaged Students. Discuss your conclusions from what you observe with these different color schemes. Note School Type is a categorical variable, but the other one is a real number.

We can see that school type is all over the place after the PCA Analysis not showing a very clear pattern. Nevertheless, we can see a pattern with economically disadvantaged students. We can practically imagine a set of clusters that differentiate the economic status of a student.

```python
#School Type
plt.scatter(df_pca['pca_0'], df_pca['pca_1'],
            c=cvec, s=50,alpha=0.5)
```

✓ 0.0s

```
<matplotlib.collections.PathCollection at 0x2a6f7fc50>
```




```python
#Economically Disadvataged
plt.scatter(df_pca['pca_0'], df_pca['pca_1'],
            c=cvec2, s=50,alpha=0.5)
```

✓ 0.0s

```
<matplotlib.collections.PathCollection at 0x2a6fc3f50>
```