In the name of God

استاد : دکتر هراتی

دانشجو : توحید حقیقی سیس

شماره دانشجویی : 830598021

موضوع : تمرین ششم

تمرین اول :

روش اول Kmeans به این صورت عمل میکند که ما در ابتدا از کل داده ها K تا به صورت رندوم انتخاب میکنیم و بعد فاصله هر نود را تا ان 3 تا حساب کرده به هر کدام نزدیک تر بود به گروه ان ملحق میشود و در مرحله بعد از هر دسته میانگین میگیریم و نقطه های جدید را ان نقطه در نظر میگیریم این کار را انقادر ادامه میدهیم که دیگر ان نقطه حرکت نکند .

در قسمت اول سوال گفته شده که از PCA برای کاهش بعد داده ها استفاده کنیم :

من این سوال رو با 2 روش حل کردم یک روش رو خودم کد kmeans رو نوشتم و در قسمت دوم با kmeans خود skitlearn انجام دادم تا ببینم درست نوشتم یا نه .

کتابخانه های استفاده شده به صورت زیر است :

```
In [78]: import numpy as np
         import pandas as pd
         from scipy.spatial import distance
         from matplotlib.pyplot import matplotlib as plt
         from sklearn.preprocessing import StandardScaler
         from sklearn.decomposition import PCA
         import seaborn as sns
```

از کتابخانه skitlearn برای PCA و همچنین استاندارد سازی داده ها استفاده کردم .

از کتابخانه Seaborn برای رسم شکل ها و نمودار های پراکندگی داده ها استفاده کردم .

و از scipy.spatial برای محاسبه فاصله بین 2 وکتور استفاده کردم .

```
import seaborn as sns

In [53]: # load dataset into Pandas DataFrame
         df = pd.read_csv('dataset.csv', names=['StockSymbol','Industry','SubIndustry','Ret2000.01','Ret2000.02','Ret2000.03','Ret2000.

In [54]:
         features = ['Ret2000.01','Ret2000.02','Ret2000.03','Ret2000.04','Ret2000.05','Ret2000.06','ret2000.07','Ret2000.08','Ret2000.
         x = df.loc[1:, features].values# Separating out the target
         y = df.loc[:,['StockSymbol','Industry','SubIndustry']].values# Standardizing the features
         x = StandardScaler().fit_transform(x)
```

در عکس بالا داده موجود در فایل ارسالی رو خوانده و تنظیمات header و تقسیم بندی داده ها به 2 دسته x , y را انجام داده ام .

```
Requirement already satisfied: pip in c:\users\tonia-pc\appdata\local\programs\python\python38-32\lib\site-packages (21.0.
1)
```

In [58]:
```
pca = PCA(n_components=2)
principalComponents = pca.fit_transform(x)
principalDf = pd.DataFrame(data = principalComponents
             , columns = ['column 1', 'column 2'])
```

In [59]:
```
principalDf
```

Out[59]:

	column 1	column 2
0	-0.651667	0.381495
1	2.725320	1.122745
2	1.540638	-0.046226
3	-2.376723	-2.970485
4	-2.165088	-1.965562
...	...	...
1153	-3.196137	-2.225590
1154	-0.225759	0.799638
1155	0.826452	2.438970
1156	5.516217	-0.731543
1157	0.903232	2.526355

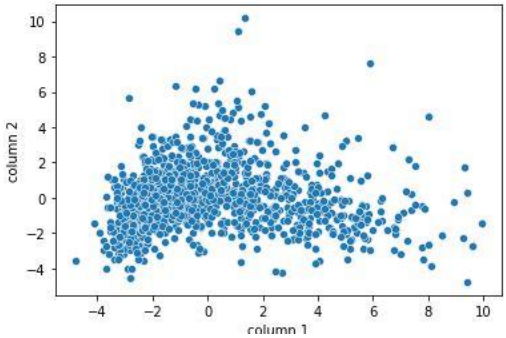
1158 rows × 2 columns

در این قسمت داده های تقسیم شده از مرحله قبل را به الگوریتم pca داده تا تبدیل به 2 بعد کند و استفاده از ان راحت تر شود .

خروجی این مرحله نیز در زیر ان نمایش داده شده است .

In [73]:
```
sns.scatterplot(data = principalDf, x = "column 1", y = "column 2",palette = "coolwarm_r")
```

Out[73]:
```
<AxesSubplot:xlabel='column 1', ylabel='column 2'>
```



پراکندگی ان را در 2 بعد به صورت بالا هست و هنوز دسته بندی نشده است و داده های خام ان است .

الگوریتم kmeans به صورت زیر نوشته شده است :

```
In [82]: def kmeans(X,k=6,max_iterations=100):
             '''
             X: multidimensional data
             k: number of clusters
             max_iterations: number of repetitions before clusters are established

             Steps:
             1. Convert data to numpy aray
             2. Pick indices of k random point without replacement
             3. Find class (P) of each data point using euclidean distance
             4. Stop when max_iteration are reached of P matrix doesn't change

             Return:
             np.array: containg class of each data point
             '''
             if isinstance(X, pd.DataFrame):X = X.values
             idx = np.random.choice(len(X), k, replace=False)
             centroids = X[idx, :]
             P = np.argmin(distance.cdist(X, centroids, 'euclidean'),axis=1)
             for _ in range(max_iterations):
                 centroids = np.vstack([X[P==i,:].mean(axis=0) for i in range(k)])
                 tmp = np.argmin(distance.cdist(X, centroids, 'euclidean'),axis=1)
                 if np.array_equal(P,tmp):break
                 P = tmp
             return P
```

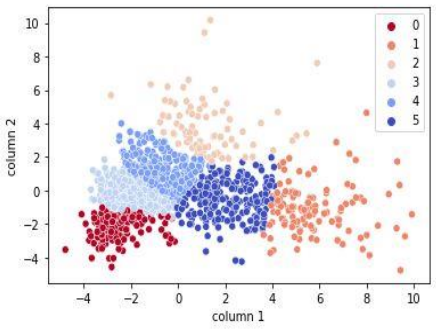
توضیحات ان نیز روی الگوریتم قرار دادم .

خروجی این مرحله را رسم کرده و به صورت زیر در امده است :

```
In [86]: # Plot clusters - this is done by colour coding the data points according to which cluster the data point belongs to
         sns.scatterplot(data=principalDf, x="column 1", y="column 2", hue= p, palette = "coolwarm_r")
         centers = km.cluster_centers_# Plot centers
         plt.scatter(centers[:, 0], centers[:, 1], c='black', s=200, alpha = 0.6);
         plt.xlabel("column 1")
         plt.ylabel("column 2")
```

```
---------------------------------------------------------------------------
AttributeError                            Traceback (most recent call last)
<ipython-input-86-785df8dd9730> in <module>
      2 sns.scatterplot(data=principalDf, x="column 1", y="column 2", hue= p, palette = "coolwarm_r")
      3 centers = km.cluster_centers_# Plot centers
----> 4 plt.scatter(centers[:, 0], centers[:, 1], c='black', s=200, alpha = 0.6);
      5 plt.xlabel("column 1")
      6 plt.ylabel("column 2")

AttributeError: module 'matplotlib' has no attribute 'scatter'
```



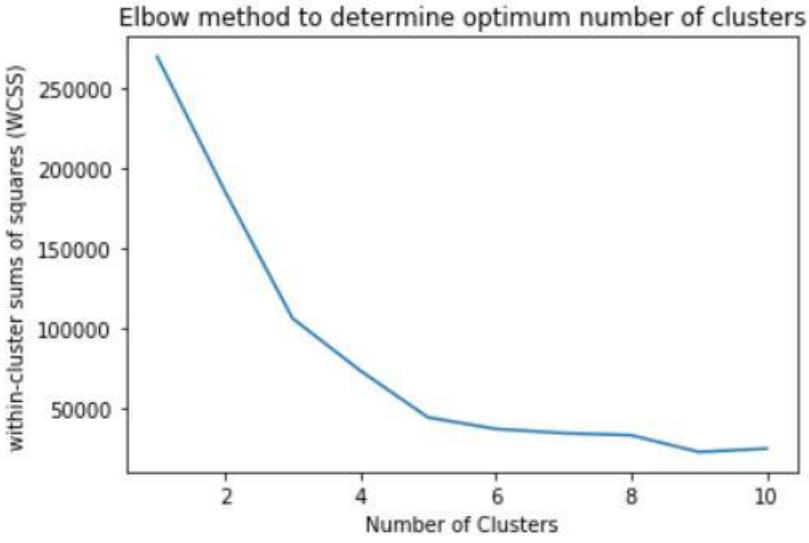
من برای اینکه بعد از دسته بندی داده ها بتوانم از روی ان سبد سهام ارایه دهم میتوانم از هر دسته یک مورد را ارائه کنم تا از هر دسته سهم یکی داشته باشید .

پس ارائه مورد به ازای هر دسته خواهد بود برای مثال اینجا که 6 دسته داریم میتوانیم از هر دسته یکی را انتخاب و برای خرید پیشنهاد دهم .

خروجی با خود الگوریتم kmeans و skilearn در کد ها وجو دارد و هر دو جواب برابر برمیگردانند .

اما برای انتخاب بهترین k چه کار میکنیم برای اینکار الگوریتم را با K های مختلف اجرا میکنیم

و مجموع فاصله های هر مرحله را محاسبه میکنیم از یک جایی به بعد فاصله خیلی کم کاهش میبابد و دیگر تغییر نمیکند .



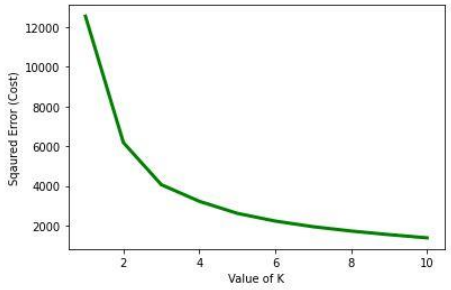
اما برای مثال ما این الگوریتم را با kهای مختلف که انجام دادم به صورت زیر انجام شد .

```
In [109]: import matplotlib.pyplot as plt
cost =[]
for i in range(1, 11):
    KM = KMeans(n_clusters = i, max_iter = 500)
    KM.fit(principalDf)

    # calculates squared error
    # for the clustered points
    cost.append(KM.inertia_)

# plot the cost against K values
plt.plot(range(1, 11), cost, color ='g', linewidth ='3')
plt.xlabel("Value of K")
plt.ylabel("Sqaured Error (Cost)")
plt.show() # clear the plot

# the point of the elbow is the
# most optimal value for choosing k
```



و نمودار انتخاب k ما به صورت بالا خواهد بود پس همانطور که میبینید از k=3 به بعد تغییر چندانی نداشته و میتوان بهترین k را 3 گرفت .

قسمت دوم این الگوریتم BFR است .

این الگوریتم مراحل انجاش به این صورت خواهد بود خیلی شبیه kmeans است فقط برای داده های بزرگ طراحی شده است در این روش به جای این که بعد از یه بار که دسته بندی کردیم مرکز را تغییر دهیم در هر بار ورود داده و دسته بندی مرکز را تغییر میدهیم .

این تغییر باعث میشود که دیگر نیازی به نگه داشتن این همه داده در رم نباشیم و برای داده هایی که در حال تغییر هستند و رفته رفته بزرگ میشوند خیلی مفید است .

در این روش ان قدر داده که در حافظه جا میشود را با الگوریتم kmeans به دست می آوریم و داده های بعدی که می آید را با این روش محاسبه میکنیم .

در این روش 3 دسته کلی داریم :

- The discard set
- The Compressed set

- The retain set

تمرین دوم :

مقاله ای که خواندم مقاله زیر است :

- **Clustering Approaches for Financial Data Analysis: a Survey**

در این مقاله گفته شده که انالیز داده های مالی رفته رفته اهمیتش در بازار سرمایه بیشتر میشود. شرکت از از عملیات روزانه مشتریان خود داده های زیادی را ذخیره میکنند و انتظار دارند با بررسی و تحلیل این داده ها الگوهایی از رفتار و عملکرد مشتریان خود پیدا کنند .

تصمیمات معقولی در برابر مشتریان خود بگیرند و روش های مختلفی برای این کار ایجاد شده است که خوشه بندی به عنوان یکی از این روش ها مورد توجه تحلیل گران قرار گرفته شده است .

در این مقاله الگوریتم های خوشه بندی های مختلفی را بررسی میکنیم و روش های تجزیه و تحلیل داده های مالی ان ها را نیز بررسی میکنیم .

این روش ها برای درک ساختار داخلی داده های مالی به کار میرود .

فرایند تصمیم گیری یکی از چالش های امروزه شرکت های دارای داده مالی هستند داده های ان ها روز به روز در حال افزایش است و تحلیل ان هر روز مشکل و مشکل تر میشود بنابراین تبدیل به یک چالش جهانی شده است و محققان نیز الگوریتم ها و روش های مختلفی توانسته اند پیدا کنند تا به اندازه ای به این زمینه کمک کرده باشند .

تکنیک های دیتا ماینینگ زیادی برای حل این چالش مطرح شده اند که به عنوان مثال درخت تصمیم و یادگیری مرتبه اول در انتخاب سهم استفاده میشوند و تکنیک های شبکه های عصبی و SVM برای پیش بینی ورشکستگی و موارد دیگر به کار میرود .

دسته بندی نزدیک ترین همسایه برای تشخیص تقلب در شبکه و همچنین کاربران میتوانند از این روش ها برای تحلیل سری های زمانی در داده های مالی نیز استفاده کنند .

در این مقاله الگوریتم های مختلف خوشه بندی بر روی داده های مالی در حوضه های کشف تقلب در کارت های اعتباری و بازار سهام و سرمایه گذاری معاملات را بررسی خواهیم کرد و همچنین معایب و مزایای ان ها را نیز بررسی میکنیم .

تحلیل مشتریان و اطلاعات ان ها و دسته بندی ان مشتریان برای بررسی موارد زیر از روی دیتاست این داده ها :

- **درک عملکرد تجاری**
- **ایجاد بازاریابی جدید ابتکارات**
- **تجزیه و تحلیل خطرات**
- **تجدیدنظر در سیاست مشتریان شرکت**
- **پیش بینی پرداخت وام**
- **تجزیه و تحلیل سیاست اعتبار مشتری**

- **بازاریابی و مراقبت از مشتری**

روش های بررسی مشتریان به Clustering و Classification تقسیم میشود .

و روشهای دیگری نیز وجود دارد مانند :

- Optimization
- regression
- simulation

به عنوان مثال ، انتخاب نمونه کارها ، مدیریت ریسک و مدیریت بدهی دارایی می تواند استفاده کند تکنیک های مختلف بهینه سازی مانند الگوریتم های ژنتیک ، برنامه نویسی پویا ، یادگیری تقویت ، و غیره ، علاوه بر این ، رگرسیون خطی و رگرسیون روشهای رایج در حوزه پیش بینی مالی ، قیمت گذاری گزینه و پیش بینی سهام.