توضیحات پروژه:
این پروژه شامل اطلاعات خروجی (فیچرهای استخراج شده) از پردازش چندین عکس در ۱۵ گروه متفاوت می‌باشد.
۱۰ درصد داده‌ها به عنوان validation و باقی آنها در test قرار گرفته اند.
با توجه به گفته صورت پروژه، الگوریتم‌های K-means، Mean-shift و Agglomerative بر این داده‌ها امتحان شد که خروجی و توضیحات هریک را در ذیل مشاهده خواهیم کرد.

الگوریتم K-means:
این الگوریتم خوشه بندی برای انجام عملیات‌های خود ابتدا k تا مرکز به صورت رندوم در نظر گرفته و براساس فاصله نقاط دیتاست به هر یک از این k مرکز، آنها را دسته بندی می‌کند. سپس مرکز هر یک از دسته ها محاسبه شده و در صورت عدم تطابق مرکز دسته قبلی و فعلی، مرکز دسته آپدیت شده و عناصر دوباره بر اساس مرکز جدید سنجیده می‌شوند.
این الگوریتم تا زمانی که هیچ نقطه‌ای دسته آن در یک دور اجرا تغییر نکند ادامه پیدا می‌کند. که البته ما threshold ای هم در نظر میگیریم که در صورتی که قبل از آن به بهترین حالت دسته بندی نرسیده بودیم تا به اندازه threshold تکرار الگوریتم را ادامه دهیم.

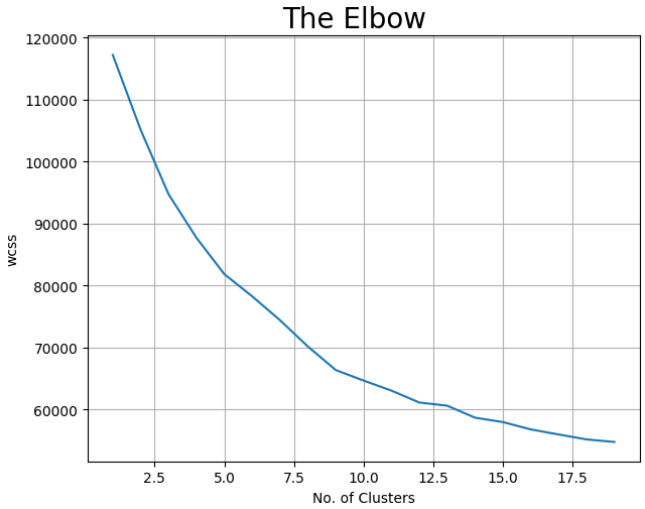
الگوریتم:

```
def k_mean(X ):
    km = KMeans(n_clusters=15,
                init='k-means++',
                max_iter=300,
                n_init=10,
                algorithm='elkan',
                tol=0.001)
    return km.fit(X)
```

همانطور که مشاهده می‌کنید این الگوریتم شامل پارامترهای مختلفی می‌باشد.

- n_cluster: تعداد کلاسترها را مشخص می‌کند. (تعداد k)
- init: متد مورد نظر برای انتخاب مرکز دسته‌های اولیه که حالت k-means++ به صورت تقریبا هوشمندی به انتخاب مراکز دسته میپردازد.
- max_iter: ماکزیمم تعداد دفعاتی که برای یک دور این الگوریتم تکرار می‌شود
- و ...

ابتدا برای پیدا کردن تعداد k مناسب میتوانیم شکل زیر را رسم کنیم:



همانطور که می‌بینید، شکستگی بیشتر بین ۱۲.۵ تا ۱۵ است. به همین خاطر من با تعداد k=13,14,15 این الگوریتم را بر داده‌های validation امتحان کردم که نتیجه معیار purity و gini index برای هر یک به شرح زیر می‌باشد(اعداد به درصد هستند). البته لازم به ذکر است که در این پروژه من به جای الگوریتم pure gini index از adjusted_rand_score استفاده کردم که حالت بهینه سازی شده gini index می‌باشد.

K	Purity	Gini index
13	87.91	80.91
14	90.93	76.64
15	90.93	63.82

همانطور که می‌بینید از نظر معیار purity تقریبا باهم برابر اند و از نظر gini index حالت k=13 درستی بیشتری داشته است. به همین خاطر ما با استفاده از همین عدد الگوریتم را برای داده‌های test استفاده می‌کنیم. با تغییر سایر پارامترها مثل اینکه init = ***random*** قرار دهیم تنها باعث عدم تغییر یا بدتر شدن مقدارهای سنجش ما می‌شدند و به همین خاطر فکر می‌کنم همین پارامترهای ست شده کافی باشند.

با اعمال الگوریتم بر داده‌های test خروجی‌های زیر حاصل شد . (جهت مقایسه، این داده‌ها را با k=14,15 نیز امتحان کردم.)

K	Purity	Gini index
13	92.76	85.08
14	91.35	65.41
15	95.39	81.52

همانطور که می‌بینید این الگوریتم با k=13 تعداد جواب درست بیشتری داشته است (gini index = 85.08) از آنجایی که purity با افزایش دسته‌ها افزایش می‌یابد و gini index درصد تعداد جواب‌های درست را بررسی می‌کند در این جا به عدد حاصل از معیار gini index بها بیشتری داده‌ام.

الگوریتم Mean-shift:
ابتدا با شروع از یک نقطه و با در نظر گرفتن شعاع مشخصی دسته‌ای از داده‌ها را در نظر می‌گیریم. درون هر دسته میانگینی از عناصر موجود در آن دسته گرفته و به عنوان مرکز جدید دسته آن را در نظر می‌گیریم. سپس در صورت عدم تطابق مرکز دسته قبلی و فعلی، مرکز دسته آپدیت می‌شود و پس از آن با توجه به مرکز جدید دوباره دسته بندی را آغاز می‌کنیم.
برتری این الگوریتم نسبت به قبلی آن است که نیازی نیست تعداد دسته‌ها را مشخص کنیم.

الگوریتم:

```
def mean_shift(X):
    return MeanShift(bandwidth=15).fit(X)
```

این الگوریتم پارامترهای مختلفی را داراست که در این جا فقط از bandwidth استفاده کردم که سایز پنجره را مشخص می‌کند.
خروجی معیارهای purity و gini index بر روی داده‌های validation به صورت زیر می‌باشد.

K	Purity	Gini index
10	100.0	3.70
15	98.18	77.68
25	25.98	1.23

که همانطور که مشاهده می‌کنید بهترین مقدار حدودا همان ۱۵ می‌باشد.

نتایج استفاده از الگوریتم بر داده‌های تست:

K	Purity	Gini index
15	99.49	85.77

الگوریتم Agglomerative:
این الگوریتم به صورت سلسله مراتبی از پایین به بالا ایجاد می‌شود به این معنا که ابتدا همه داده‌ها به صورت تکی به عنوان کلاستر در نظر گرفته می‌شوند و رفته رفته باهم ترکیب شده و کلاسترهای بزرگ‌تر را تشکیل می‌دهند.

الگوریتم:

```
def agglomerative_clustering(X, link):
    return AgglomerativeClustering(n_clusters=15, linkage=link).fit_predict(X)
```

پارامترهای این الگوریتم:

- n_clusters: تعداد کلاسترهایی که باید پیدا شوند را مشخص می‌کند.
- linkage: نوع فاصله گذاری و تشخیص شباهت دو کلاستر را مشخص می‌کند که شامل حالت‌های "ward", "complete", "average", "single" می‌شود که توضیحات آن‌ها را در درس داشته‌ایم.
- و ...

خروجی معیارهای purity و gini index بر روی داده‌های validation برای مقادیر مختلف linkage به صورت زیر می‌باشد. خروجی این جدول به ازای n_cluster=15 ایجاد شده است.

Link	Purity	Gini index
Single	30.21	4.23
Complete	91.23	88.80
Average	63.14	51.24
Ward	94.25	85.26

همانطور که مشاهده می‌کنید link = complete, ward تقریبا نتایج مشابه‌ای را ایجاد کرده اند. و تقریبا نتیجه complete بهتر است.

خروجی معیارهای purity و gini index بر روی داده‌های test برای مقادیر مختلف linkage به صورت زیر می‌باشد.

Link	Purity	Gini index
Complete	89.33	85.92
Ward	98.72	60.82

که همانطور که گفته شد complete نتیجه بهتری داشت.

مقایسه عملکرد الگوریتم ها:
بر اساس معیار gini index تقریبا همه انها در یک سطح عمل کرده اند و مقدار این عدد در الگوریتم agglomerative اندکی بیشتر می‌باشد.

استفاده از دو الگوریتم K-means و Mean-shift:
در این قسمت ابتدا با استفاده از الگوریتم Mean-shift مراکز دسته‌ها را بدست اوردم و سپس آنها را به عنوان دسته‌های اغازین به الگوریتم K-means دادم. که البته بهبود خاصی حاصل نشد.

اجرا بر داده‌های validation:

Purity	Gini index
98.18	77.68

Purity	Gini index
98.82	54.04