به نام خدا

دانشگاه تهران

دانشکده مهندسی برق و کامپیوتر

درس یادگیری ماشین

تمرین پنجم

نام و نام خانوادگی	میلاد محمدی
شماره دانشجویی	810100462
تاریخ ارسال گزارش	1401/11/16

سوال اول

قسمت 1) خیر یکی نیستند. Model selection در واقع به انتخاب مدل مناسب از بین مدل‌های موجود با توجه به معیارهای خاصی که به آن‌ها نیاز داریم اشاره دارد ولی model assessment به ارزیابی یک مدل براساس معیارهایی مانند accuracy، precision و ... اشاره دارد.

قسمت 2) یکی از روش‌های انتخاب مدل با کمترین خطای generalization، استفاده از cross validation است. در این روش داده‌ها را به چندین دسته تقسیم می‌کنیم. آموزش مدل بر روی برخی از دسته‌ها و سپس ارزیابی مدل را روی دسته‌های دیگر انجام می‌دهیم. این فرآیند چندین بار با ترکیب‌های مختلف داده‌های آموزشی و تست تکرار می‌شود تا تخمینی از خطای generalization مدل ارائه شود. cross validation روش مناسبی است که برای جلوگیری از overfitting کارآمد است.

قسمت 3)

Leave-one-out cross-validation: این روش شبیه cross-validation k-fold است اما به جای اینکه داده‌ها را به k fold تقسیم کنیم، مدل را روی همه نمونه‌ها به جز یک نمونه آموزش می‌دهیم و آن را روی نمونه تست آزمایش می‌کنیم. این فرآیند برای هر نمونه تکرار می‌شود و تخمین قوی‌تری از عملکرد مدل ارائه می‌دهد.

Bootstrap: روش بوت‌استرپ یک تکنیک نمونه‌گیری مجدد است که برای تخمین آمار یک جمعیت با نمونه‌برداری از مجموعه داده با جایگزینی استفاده می‌شود. می‌توانیم از بوت‌استرپ برای تخمین تغییرپذیری عملکرد مدل با چندین بار نمونه‌برداری مجدد از داده‌ها و آموزش مدل بر روی هر مجموعه داده نمونه‌برداری شده استفاده کنیم.

مدل‌های ساده: وقتی حجم داده‌ها کم است، آموزش و ارزیابی مدل‌های ساده، مانند رگرسیون خطی یا درخت‌های تصمیم، اغلب آسان‌تر است، زیرا پارامترهای کمتری برای تخمین دارند و احتمالش کم است که دچار overfitting شوند.

Regularization: روشی است که هدف آن کاهش پیچیدگی مدل با افزودن یک عبارت جریمه به تابع هزینه است. این عبارت مدل را از تطبیق نویز در داده‌ها دور می‌کند که می‌تواند به ویژه زمانی که مقدار داده کم است مفید باشد.

قسمت 4)

یکی از معیارهای رایج مورد استفاده برای ارزیابی مدل‌ها accuracy است. معیار accuracy تعداد دفعاتی را که مدل برچسب یک نقطه داده معین را به درستی پیش‌بینی می‌کند، اندازه‌گیری می‌کند و معمولاً برای مسائل طبقه‌بندی استفاده می‌شود.

معیار دیگری که می‌تواند برای ارزیابی مدل‌ها مورد استفاده قرار گیرد، امتیاز F1 است. امتیاز F1 میانگین هارمونیک precision و recall است که در آن precision، نسبت پیش‌بینی‌های مثبت واقعی از همه پیش‌بینی‌های مثبت است و recall نسبت پیش‌بینی‌های مثبت واقعی از همه نمونه‌های مثبت واقعی است. امتیاز F1 معیار خوبی برای استفاده زمانی است که کلاس‌های مجموعه داده نامتعادل هستند.

سوال دوم

قسمت 1) هدف از انتخاب ویژگی، انتخاب زیرمجموعه‌ای از ویژگی‌ها از مجموعه بزرگ‌تری از ویژگی‌ها، به منظور بهبود عملکرد و تفسیرپذیری یک مدل یادگیری ماشینی است. با حذف ویژگی‌های نامربوط، زائد یا نویزی، انتخاب ویژگی می‌تواند ابعاد داده‌ها را کاهش داده و از overfitting جلوگیری کند در حالی که همچنان اطلاعات مهم مورد نیاز برای پیش‌بینی‌های دقیق را حفظ می‌کند. علاوه بر این، می‌تواند به شناسایی مهم‌ترین ویژگی‌هایی که پیش‌بینی‌های مدل بیشتر بر اساس آن‌ها انجام می‌شود، کمک کند. دادن همه ویژگی‌ها به یک الگوریتم یادگیری ماشین برای تصمیم‌گیری در مورد انتخاب ویژگی‌ها امکان‌پذیر است اما می‌تواند از نظر محاسباتی گران باشد و لزوماً منجر به عملکرد بهتر نشود. برخی از الگوریتم‌ها، مانند Decision tree و random forest، دارای روش‌های انتخاب ویژگی داخلی هستند که می‌توانند مهم‌ترین ویژگی‌ها را با استفاده از فرآیندی به نام اهمیت ویژگی شناسایی کنند. با این حال، این روش‌ها ممکن است همیشه دقیق نباشند و می‌توانند تحت تأثیر هایپر پارامترهای خاص الگوریتم یا ساختار داده‌ها قرار گیرند. علاوه بر این، بسیاری از الگوریتم‌های یادگیری ماشین روش‌های انتخاب ویژگی داخلی ندارند و ممکن است نتوانند تعداد زیادی از ویژگی‌ها را به طور موثر مدیریت کنند.

قسمت 2) امتیاز فیشر روشی برای انتخاب ویژگی است که مبتنی بر این ایده است که ویژگی‌هایی که برای طبقه‌بندی مفیدتر هستند باید تفاوت زیادی در مقادیر میانگین بین کلاس‌های مختلف و واریانس کمی در هر کلاس داشته باشند. امتیاز فیشر از یک ویژگی به عنوان نسبت واریانس بین کلاس به واریانس درون کلاس محاسبه می‌شود. برای دو کلاس a و b به صورت زیر محاسبه می‌شود:

$$fisher's\ score = \frac{(\mu_a - \mu_b)^2}{s_a^2 + s_b^2}$$

ویژگی‌هایی با امتیاز فیشر بالاتر برای طبقه‌بندی مرتبط‌تر در نظر گرفته می‌شوند و بنابراین برای مدل نهایی انتخاب می‌شوند. امتیاز فیشر یک روش خطی است به این معنی که فرض می‌کند که رابطه بین ویژگی‌ها و متغیر هدف خطی است ولی این فرض ممکن است همیشه برقرار نباشد.

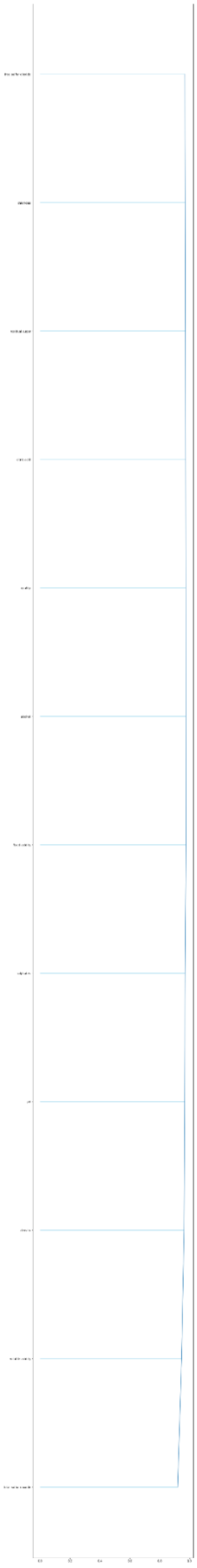
سوال سوم

قسمت 1) Forward selection یک الگوریتم انتخاب ویژگی است که در یادگیری ماشین و مدل‌سازی آماری استفاده می‌شود. یک الگوریتم حریصانه است که با مجموعه‌ای از ویژگی‌ها شروع می‌شود و هر بار یک ویژگی را اضافه می‌کند تا زمانی که مجموعه‌ای از ویژگی‌هایی که متغیر پاسخ را به‌طور بهینه توضیح می‌دهند به دست آید. ویژگی که باید در هر مرحله اضافه شود، همان ویژگی است که بیشترین افزایش را در عملکرد مدل دارد و تا جایی که دقت مدل افزایش می‌یابد، ویژگی را اضافه می‌کنیم و در صورت کم شدن دقت الگوریتم متوقف می‌شود. خروجی الگوریتم در صفحه 5 نشان داده شده است.

پیاده‌سازی:

```
def seq_forward_selection(X_train, X_test, y_train, y_test):
    columns = list(X_train.columns.copy())
    length = len(columns)


    selected_cols = []

    answer = []
    for i in range(length):

        preds = []
        for col in columns:
            try:
                selected_cols.append(col)
                X_train_new = X_train[selected_cols]
                clf = GaussianNB()
                clf.fit(X_train_new, y_train)
                pred_score = clf.score(X_test[selected_cols] , y_test)
                preds.append(pred_score)
                selected_cols.pop()
            except:
                pass

        best_index = np.argmax(preds)
        next_best_feature = columns.pop(best_index)
        answer.append((next_best_feature , preds[best_index] ))
        selected_cols.append(next_best_feature)
#         print(answer[-1])
#         print(selected_cols)

    return answer
```

شکل 1. پیاده‌سازی الگوریتم **Forward selection**

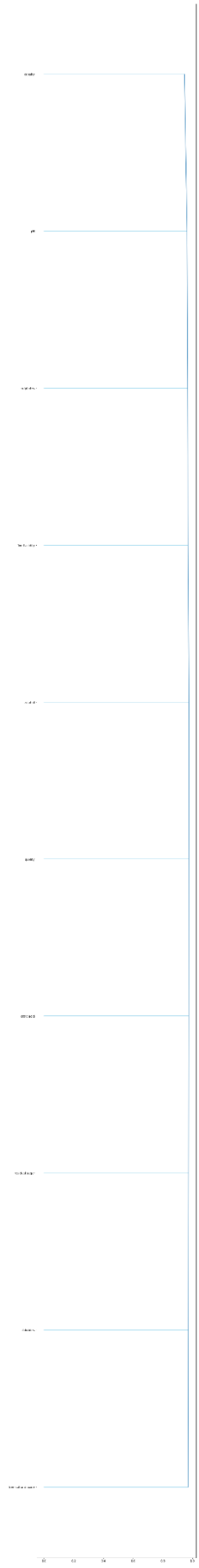


قسمت 2) Recursive Feature Elimination یک الگوریتم انتخاب ویژگی است که در یادگیری ماشین و مدل‌سازی آماری استفاده می‌شود. یک فرآیند بازگشتی است که با همه ویژگی‌ها شروع می‌شود و کم اهمیت ترین ویژگی را به طور مکرر حذف می‌کند تا زمانی که تعداد مطلوب ویژگی‌ها به دست آید. اهمیت ویژگی با استفاده از مدلی مانند رگرسیون خطی، درخت تصمیم یا ماشین‌های بردار پشتیبان تعیین می‌شود. RFE برای حذف ویژگی های اضافی یا نامربوط و بهبود عملکرد مدل مفید است. خروجی الگوریتم در صفحه 7 نشان داده شده است.

پیاده‌سازی:

```
def seq_backward_selection(X_train, X_test, y_train, y_test):
    columns = list(X_train.columns.copy())
    length = len(columns)



    answer = []
    for i in range(length - 2):
        preds = {}
        selected_cols = columns.copy()

        for col in columns:
            try:
                selected_cols.remove(col)
                X_train_new = X_train[selected_cols]
                clf = GaussianNB()
                clf.fit(X_train_new, y_train)
                pred_score = clf.score(X_test[selected_cols] , y_test)
                preds[col] = pred_score
                selected_cols.append(col)
            except:
                break
        best_to_remove = max(preds, key=preds.get)
        columns.remove(best_to_remove)
        answer.append((best_to_remove , preds[best_to_remove] ))
        print(i)
        print(answer[-1])

    return answer
```

شکل 2. پیاده‌سازی الگوریتم **Recursive Feature Elimination**



قسمت 3) روش Recursive Feature Elimination عموما روش ترجیحی است زیرا روش Forward selection به اصطلاح اثرات suppressor را ایجاد می‌کند. این اثرات suppressor زمانی رخ می‌دهند که پیش‌بینی‌کننده‌ها تنها زمانی مهم باشند که پیش‌بینی‌کننده دیگری ثابت نگه داشته شود. این بدان معنی است که برخی از پیش بینی کننده‌ها که خوب نیستند در تکرار قبلی انتخاب می‌شوند و تا پایان حذف نمی‌شوند. به نظر می‌رسد Forward selection سریعتر از Recursive Feature Elimination عمل می‌کند. اما عملکرد بدتری دارد.

سوال چهارم

*** ذکر این نکته لازم است که تصاویر به 128 * 128 reshape شده‌اند. چون در حالت 256*256 برای به دست آوردن مقادیر ویژه و بردارهای ویژه به خطای زیر برخورد کردیم و برایتان قبلا ایمیل نیز کردم.

```
cov.shape
```

```
(65536, 65536)
```

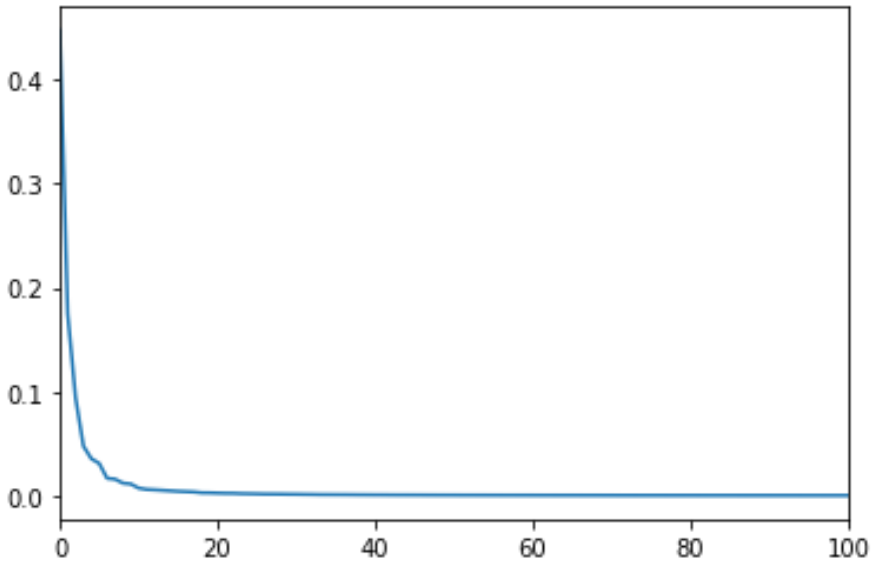
```
eigenvalues, eigenvectors = np.linalg.eig(cov)
```

```
---------------------------------------------------------------------------
MemoryError                               Traceback (most recent call last)
~\AppData\Local\Temp/ipykernel_12132/1914764623.py in <module>
----> 1 eigenvalues, eigenvectors = np.linalg.eig(cov)

<__array_function__ internals> in eig(*args, **kwargs)

~\anaconda3\lib\site-packages\numpy\linalg\linalg.py in eig(a)
   1321         _raise_linalgerror_eigenvalues_nonconvergence)
   1322     signature = 'D->DD' if isComplexType(t) else 'd->DD'
-> 1323     w, vt = _umath_linalg.eig(a, signature=signature, extobj=extobj)
   1324
   1325     if not isComplexType(t) and all(w.imag == 0.0):

MemoryError: Unable to allocate 64.0 GiB for an array with shape (65536, 65536) and data type complex128
```

در فشرده‌سازی تصویر، PCA می‌تواند برای شناسایی مهم‌ترین ویژگی‌های یک تصویر و کاهش تعداد بیت‌های مورد نیاز برای نمایش تصویر استفاده شود. این می‌تواند منجر به کاهش قابل توجهی در اندازه فایل شود که می‌تواند به ویژه برای تصاویری که بزرگ هستند یا باید از طریق شبکه‌های آهسته منتقل شوند، مهم باشد. از نظر افزایش دقت برای کارهای مختلف، PCA می‌تواند برای بهبود عملکرد الگوریتم‌های یادگیری ماشینی که بر روی تصاویر اعمال می‌شود مفید باشد. به عنوان مثال، وظیفه‌ای مانند تشخیص چهره را در نظر بگیرید. PCA می‌تواند برای استخراج مهم ترین ویژگی‌های صورت مانند شکل چشم‌ها، بینی و دهان و نمایش این ویژگی‌ها در فضایی با ابعاد کمتر استفاده شود. این می‌تواند منجر به بهبود دقت

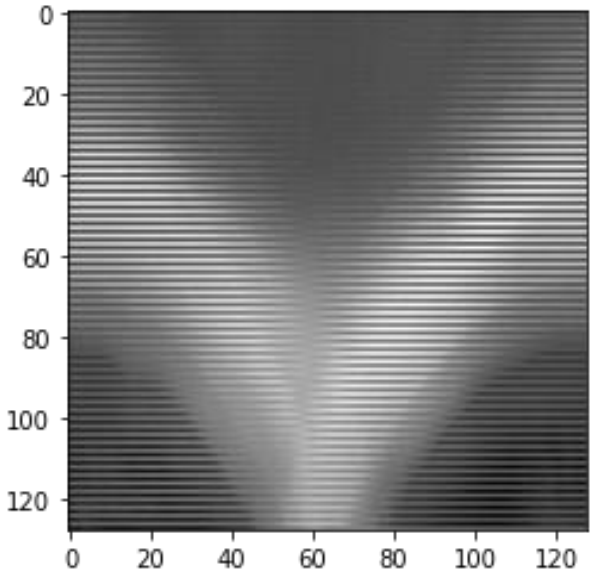
الگوریتم‌های تشخیص چهره شود، زیرا آنها به جای تلاش برای پردازش تعداد زیادی پیکسل در یک تصویر، فقط باید مهمترین اطلاعات را پردازش کنند.

قسمت 1) ما از بردارهای ویژه استفاده می‌کنیم که کمتر از یک مقدار آستانه در واریانس بیش از همه سهم دارند. برای انجام این کار، واریانس نسبی مقادیر ویژه مختلف را با استفاده از نمودار رسم می‌کنیم. می‌توانیم ببینیم 20 آستانه خوبی است، اما برای قسمت بعدی از 100 استفاده خواهیم کرد. شکل 3 نشان‌دهنده این واقعیت است.

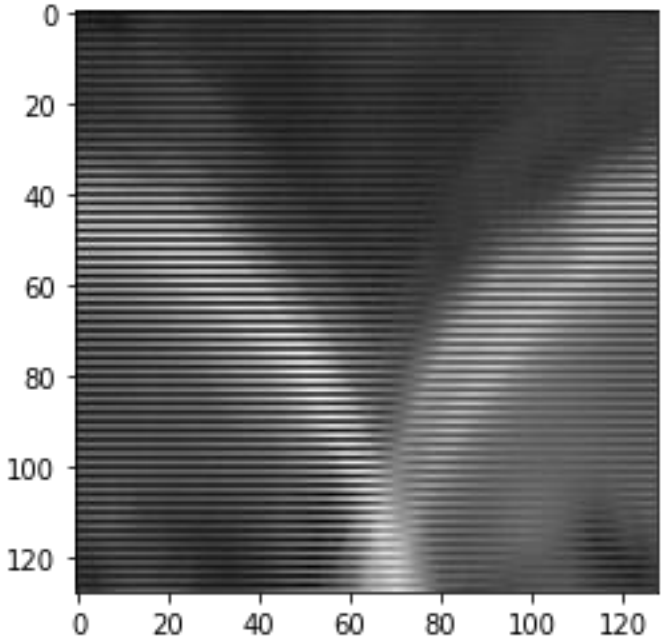


شکل 3. نمودار واریانس نسبی

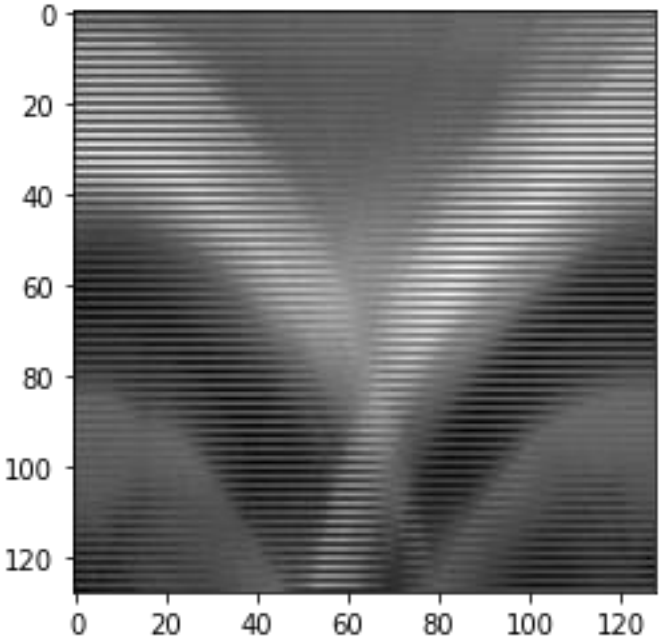
قسمت 2) این تصاویر نشان می‌دهد که هر بردار ویژه حاوی چه اطلاعاتی است. آخرین تصاویر بیشتر نویز هستند.



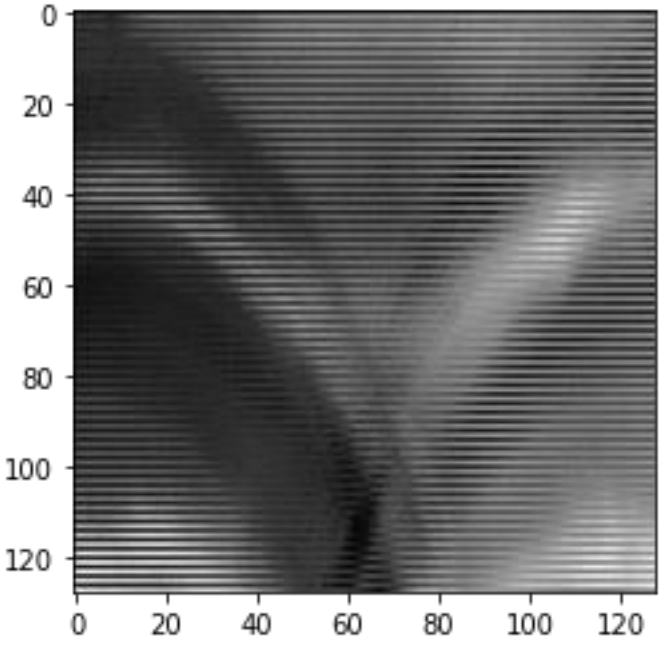
شکل 4. مقدار ویژه اول



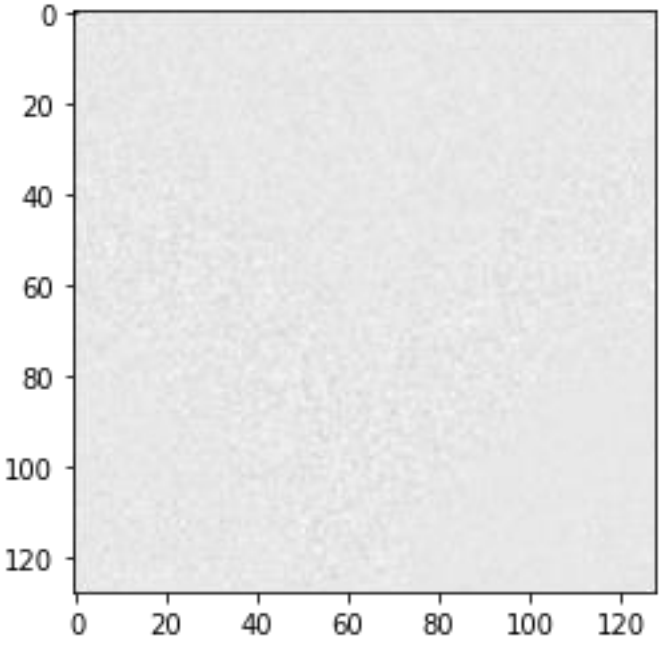
شکل 5. مقدار ویژه دوم



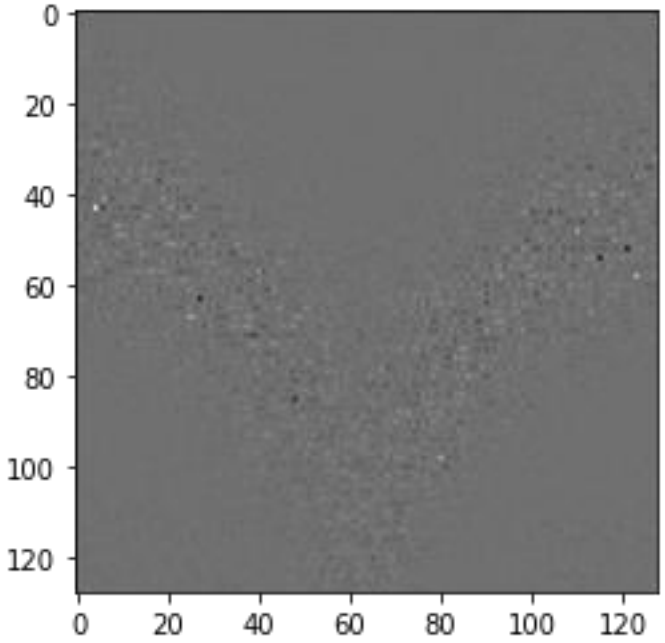
شکل 6. مقدار ویژه سوم



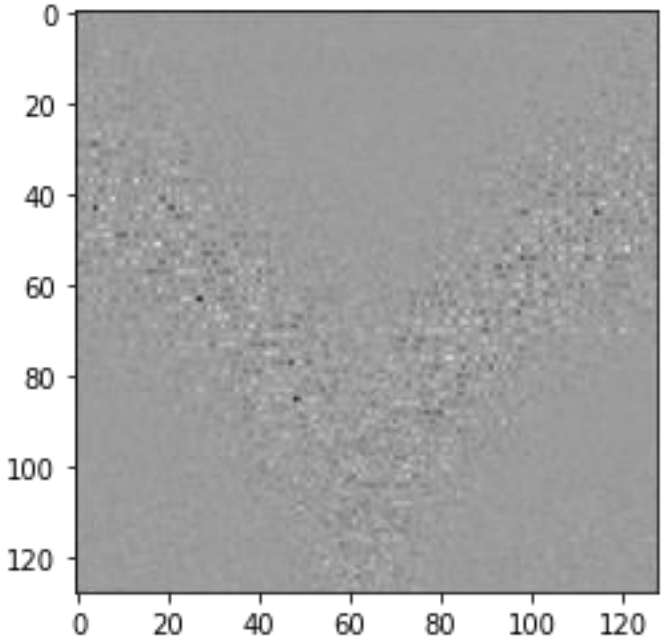
شکل 7. مقدار ویژه چهارم



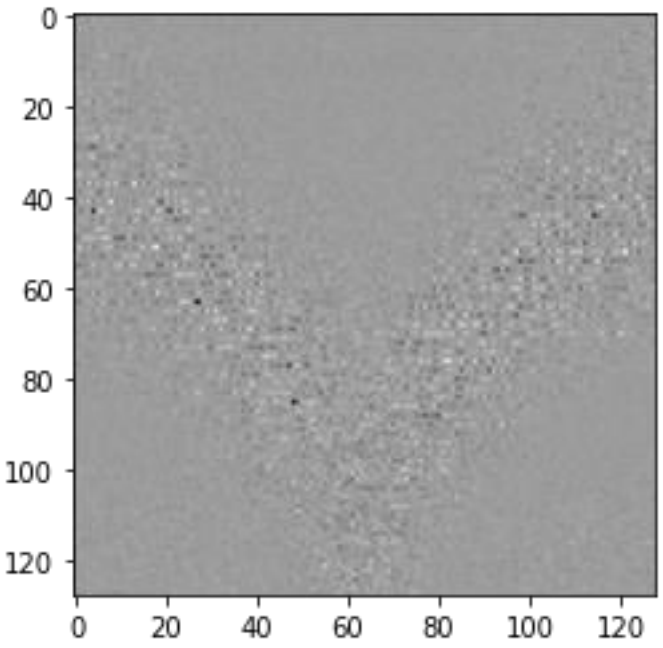
شکل 8. مقدار ویژه آخر



شکل 9. مقدار ویژه یکی مانده به آخر



شکل 10. مقدار ویژه دو تا مانده به آخر



شکل 11. مقدار ویژه سه تا مانده به آخر

قسمت 3)

K=2

```
In [23]: neigh_reduced = KNeighborsClassifier(n_neighbors=2)
         neigh_reduced.fit(new_X_train,y_train)

Out[23]: KNeighborsClassifier(n_neighbors=2)

In [24]: y_pred_reduced = neigh_reduced.predict(new_X_test)
         confusion_matrix(y_test,y_pred_reduced)

Out[24]: array([[ 0,  0,  0,  0,  0,  0],
                [ 4,  7,  0,  0,  0,  0],
                [ 2,  1,  6,  3,  0,  0],
                [ 2,  1,  0, 10,  2,  1],
                [ 2,  1,  0,  5,  2,  0],
                [ 0,  0,  1,  5,  2,  2]], dtype=int64)

In [25]: accuracy_score(y_test,y_pred_reduced)

Out[25]: 0.4576271186440678
```

K=1

In [26]:
```
neigh_reduced = KNeighborsClassifier(n_neighbors=1)
neigh_reduced.fit(new_X_train,y_train)
```

Out[26]:
```
KNeighborsClassifier(n_neighbors=1)
```

In [27]:
```
y_pred_reduced = neigh_reduced.predict(new_X_test)
```

In [28]:
```
confusion_matrix(y_test,y_pred_reduced)
```

Out[28]:
```
array([[ 0,  0,  0,  0,  0,  0],
       [ 2,  7,  0,  1,  1,  0],
       [ 2,  0,  6,  1,  2,  1],
       [ 0,  0,  0, 11,  3,  2],
       [ 0,  2,  0,  3,  3,  2],
       [ 0,  0,  1,  3,  2,  4]], dtype=int64)
```

In [29]:
```
accuracy_score(y_test,y_pred_reduced)
```

Out[29]:
```
0.5254237288135594
```

Not reduced, K=2

In [31]:
```
neigh_reduced = KNeighborsClassifier(n_neighbors=2)
neigh_reduced.fit(X_train,y_train)
```

Out[31]:
```
KNeighborsClassifier(n_neighbors=2)
```

In [32]:
```
y_pred = neigh_reduced.predict(X_test)
```

In [33]:
```
confusion_matrix(y_test,y_pred)
```

Out[33]:
```
array([[0, 0, 0, 0, 0, 0],
       [4, 7, 0, 0, 0, 0],
       [2, 2, 6, 2, 0, 0],
       [3, 1, 0, 9, 2, 1],
       [2, 1, 0, 5, 2, 0],
       [0, 0, 1, 5, 2, 2]], dtype=int64)
```

In [34]:
```
accuracy_score(y_test,y_pred)
```

Out[34]:
```
0.4406779661016949
```

Not reduced, k=1

```
In [35]: neigh_reduced = KNeighborsClassifier(n_neighbors=1)
         neigh_reduced.fit(X_train,y_train)
```

```
Out[35]: KNeighborsClassifier(n_neighbors=1)
```

```
In [36]: y_pred = neigh_reduced.predict(X_test)
```

```
In [37]: confusion_matrix(y_test,y_pred)
```

```
Out[37]: array([[ 0,  0,  0,  0,  0,  0],
                [ 2,  7,  0,  1,  1,  0],
                [ 2,  0,  7,  1,  1,  1],
                [ 0,  0,  0, 10,  3,  3],
                [ 0,  2,  0,  3,  3,  2],
                [ 0,  0,  1,  3,  2,  4]], dtype=int64)
```

```
In [38]: accuracy_score(y_test,y_pred)
```

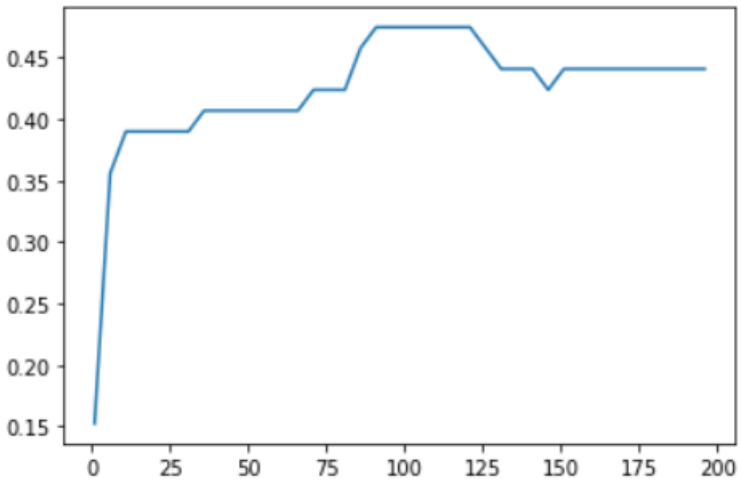
```
Out[38]: 0.5254237288135594
```

با مقایسه نتایج به دست آمده متوجه می‌شویم که کاهش ابعاد تاثیری بر دقت طبقه‌بندی نداشته است ولی مزیتی که دارد کاهش حافظه لازم برای فشرده‌سازی تصاویر است.

قسمت 4) کاهش ابعاد منجر به از دست دادن دقت تا یک آستانه خاص می‌شود، پس از آن خطای تخمین برقرار می‌شود و پس از یک نقطه از دست دادن دقت دوباره شروع می‌شود.

```
In [40]: plt.plot(axis,acc_scores)
```

```
Out[40]: [<matplotlib.lines.Line2D at 0x1864900f8b0>]
```



سوال ششم

قسمت 1)

خوشه بندی بر اساس فاصله بین نقاط همیشه درست کار نمی‌کند و تحت شرایط خاصی می‌تواند منجر به نتایج منفی شود، مانند:

1. **خوشه‌های با اندازه غیر یکنواخت**: اگر خوشه‌ها دارای اندازه‌های غیر یکنواخت باشند، ممکن است منجر به گروه‌بندی نادرست نقاط شود زیرا ممکن است فاصله‌ها به طور دقیق رابطه بین نقاط در خوشه‌های مختلف را منعکس نکنند.

2. **خوشه‌های غیرکروی**: اگر خوشه‌ها اشکال غیرکروی داشته باشند، روش‌های سنتی مبتنی بر فاصله مانند فاصله اقلیدسی ممکن است در گرفتن روابط بین نقاط موثر نباشد.

3. **نقاط پرت (Outliers)**: نقاط پرت می‌توانند به طور قابل توجهی بر نتایج خوشه بندی مبتنی بر فاصله تأثیر بگذارند، زیرا ممکن است به عنوان نقاط مرکزی در نظر گرفته شوند که منجر به گروه‌بندی نادرست سایر نقاط می‌شود.

4. **روابط غیرخطی**: اگر روابط بین نقاط غیرخطی باشد، روش‌های سنتی مبتنی بر فاصله ممکن است مناسب نباشند، زیرا روابط را خطی فرض می‌کنند.

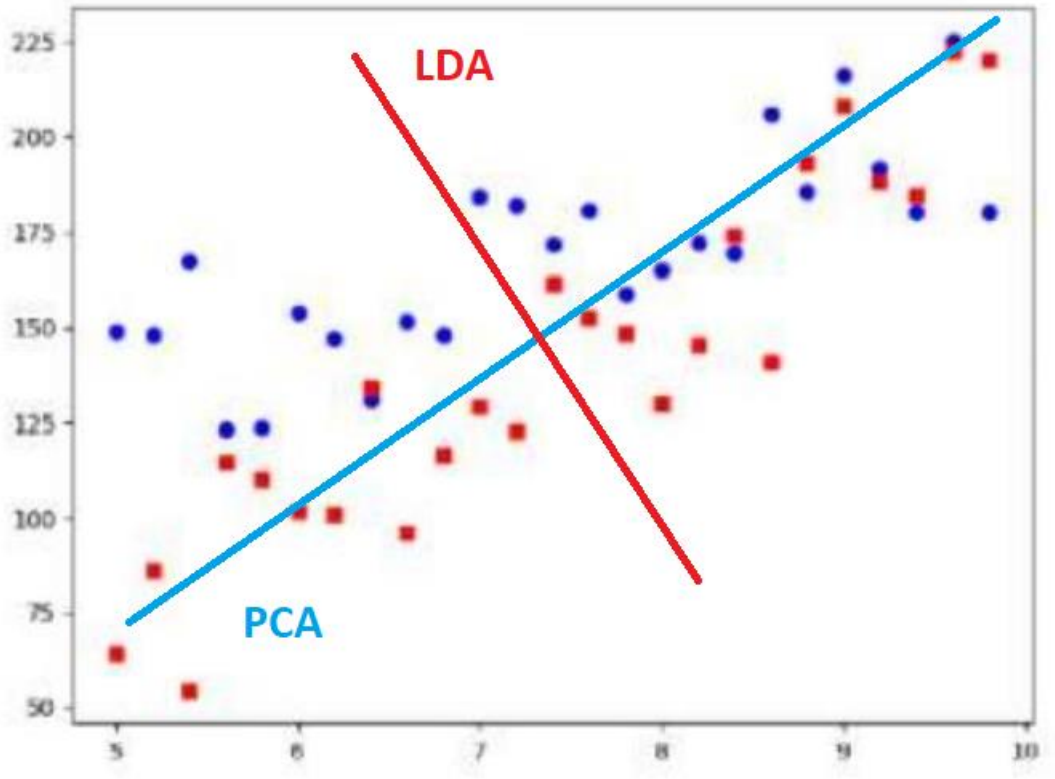
قسمت 2)

DBSCAN یک الگوریتم خوشه‌بندی مبتنی بر چگالی است که نقاطی را که به طور متراکم در فضای ویژگی‌ها در کنار هم قرار گرفته‌اند را گروه‌بندی می‌کند. با تعریف متریک فاصله و حداقل تعداد نقاط مورد نیاز برای تشکیل یک خوشه کار می‌کند. الگوریتم با انتخاب یک نقطه تصادفی و بررسی اینکه آیا حداقل نقاط در یک شعاع مشخص (Eps) از آن نقطه وجود دارد یا خیر شروع می‌شود. اگر وجود داشته باشد، این نقاط به عنوان یک خوشه در نظر گرفته می‌شوند و الگوریتم با اضافه کردن نقاط نزدیکی که در شعاع Eps هستند و همچنین حداقل نقاطی که در مجاورت خود دارند، به گسترش خوشه ادامه می‌دهد. این فرآیند تا زمانی ادامه می‌یابد که نتوان نقطه‌ای به خوشه اضافه کرد و سپس الگوریتم فرآیند را برای نقاط دیده نشده تکرار می‌کند. نقاطی که به هیچ خوشه‌ای تعلق ندارند نویز محسوب می‌شوند. الگوریتم OPTICS (Ordering Points To Identify the Clustering Structure) یک الگوریتم

خوشه‌بندی مبتنی بر چگالی است که شبیه به DBSCAN است. با این حال، در حالی که DBSCAN با گروه‌بندی مستقیم نقاطی که به یکدیگر نزدیک هستند، خوشه‌ها را شناسایی می‌کند، OPTICS یک نمودار دسترسی ایجاد می‌کند که روابط مبتنی بر چگالی بین نقاط را نشان می‌دهد و سپس خوشه‌ها را بر اساس این نمودار شناسایی می‌کند. به عبارت دیگر، OPTICS نمای کامل تری از روابط مبتنی بر چگالی بین نقاط، از جمله شکل‌ها و اندازه‌های خوشه‌ها و فواصل بین آنها ارائه می‌دهد، در حالی که DBSCAN نمای مستقیم تر، اما با جزئیات کمتری از این روابط ارائه می‌دهد.

در نتیجه، DBSCAN و OPTICS هر دو الگوریتم‌های خوشه‌بندی مبتنی بر چگالی هستند، اما در نحوه دریافت روابط مبتنی بر چگالی بین نقاط تفاوت دارند.

سوال هفتم



الگوریتم PCA براساس واریانس داده‌ها عمل می‌کند و در هر جهت که بیشترین واریانس وجود داشته باشد، بدون توجه به برچسب داده‌ها اولین کامپوننت خود را انتخاب می‌کند. ولی الگوریتم LDA براساس امتیاز فیشر عمل می‌کند و براساس اینکه scatter بیرونی داده‌ها زیاد ولی scatter درونی آن‌ها کمتر باشد خط را رسم می‌کند. در این مجموعه داده LDA عملکرد بهتری دارد چرا که با توجه به برچسب داده‌ها خطی که رسم شده تمایز بیشتری قائل می‌شود.