به نام خدا

دانشگاه صنعتی امیرکبیر (پلی تکنیک تهران)

دانشکده مهندسی کامپیوتر

یادگیری ماشین

پروژه پایانی

استاد درس: دکتر ناظرفرد

امیرحسین کاشانی

۴۰۰۱۳۱۰۷۱

amkkashani@gmail.com

نیم سال اول ۱۴۰۱-۱۴۰۲

Contents

Q1) 3

A) 3

B) 5

C) 6

D) 7

E) 8

F) 8

g) 9

Q2) 11

A,B) 11

C) 12

D) 14

E) 16

F) 16

G) 17

Q3) 19

A) 19

B) 20

C) 21

D) 22

E) 23

F) 24

G) 25

H) 26

i) 27

j) 28

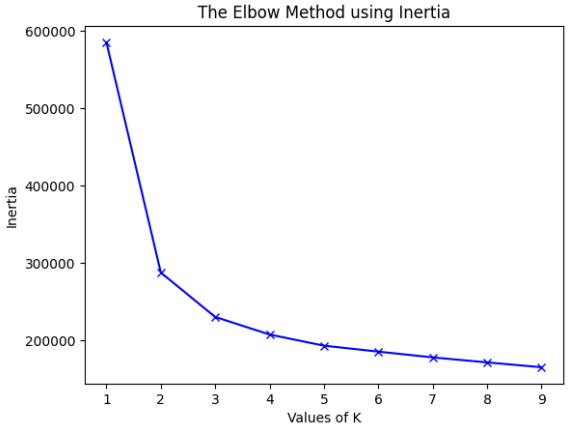
K) 29

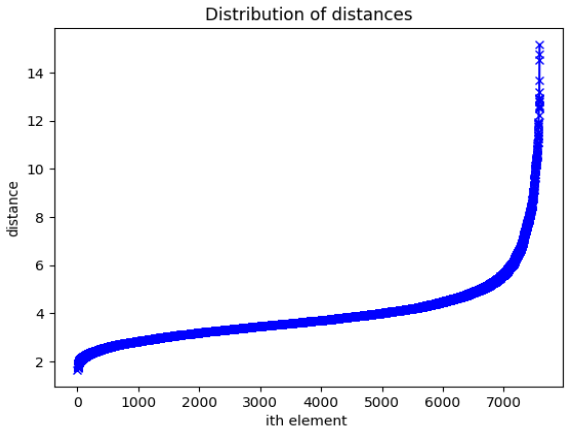
Q1)

A)

در این بخش به پیش پردازش داده‌های می‌پردازیم. دو ستون اول که شماره ایستگاه و تاریخ هستند در بخش محاسبات وارد نمی‌شوند و فقط ازتاریخ برای جدا سازی بهره گرفته شده است. در این بخش دو سطر آخر از داده نیز حذف شده اند که اطلاعات آماری ستون‌ها بوده اند و همچنین داده‌های صفر نیز که معادل NAN بوده اند با میانگین ستون خود پر شده اند.

همچنین با استفاده از elbow در k_means به تشخیص داده‌های پرت پرداخته شده است که در شکل اول نتایج اجراهای مختلف صورت گرفته و در شکل دوم فاصله نقاط(مرتب شده بر اساس فاصله) را مشاهده می‌کند.(شکل دوم با k=3 کشیده شده است، k براساس شکل اول انتخاب گردیده است)

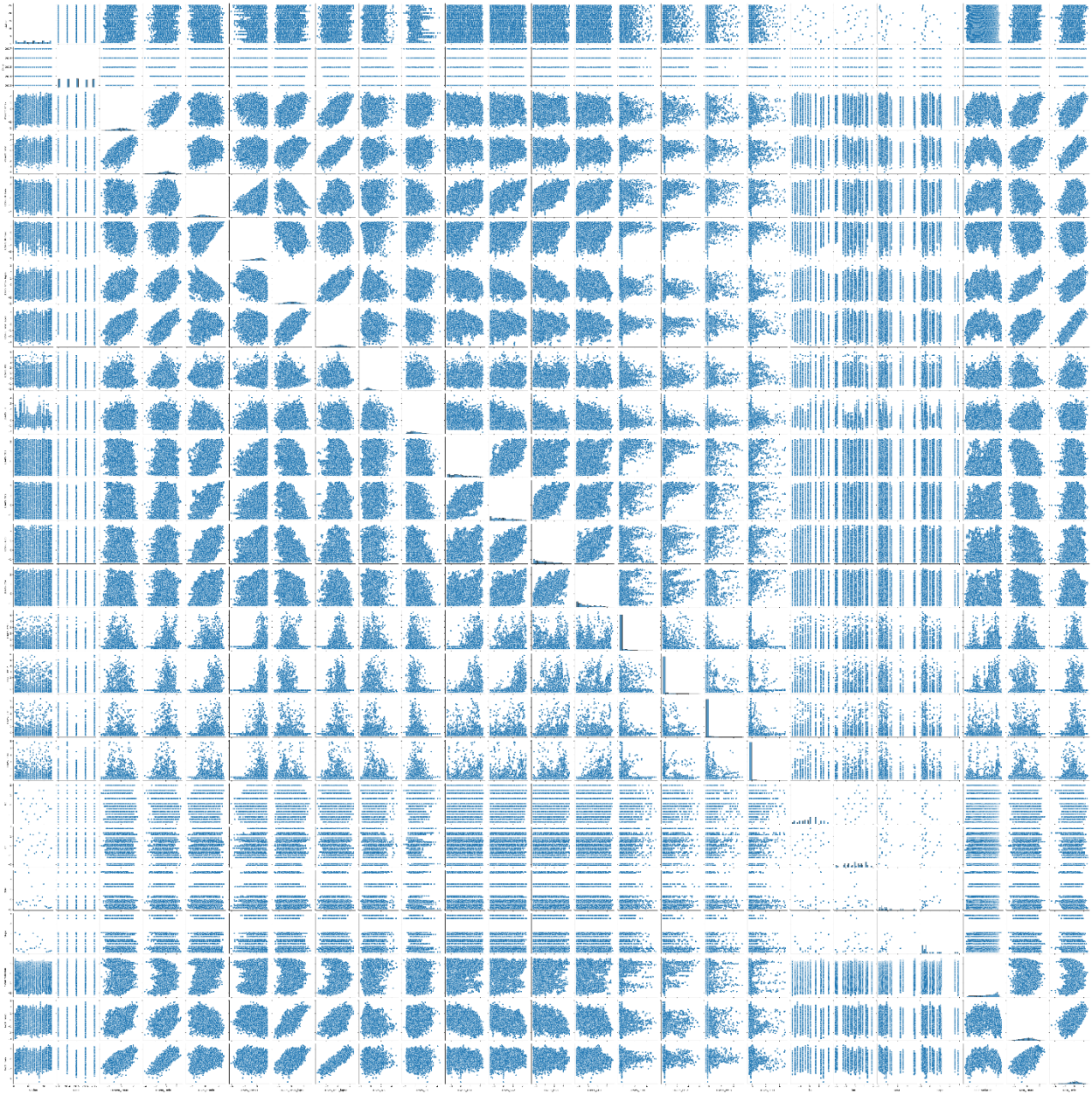




سپس با توجه به شکل دوم نقاطی که فاصله بیش از ۶ داشته اند را حذف کرده ایم.

در نهایت نیز بر اساس تاریخ داده‌ها به دو بخش train و test تقسیم گردیده اند.

برای اطمینان بیشتر و انتخاب مناسب ویژگی‌ها pairplot مربوط به داده‌های سوال نیز ترسیم شده است که به مورد مشکوکی در رابطه با وابستگی خطی بیش از حد بین ویژگی‌ها دیده نشده است و درنتیجه همه ستون‌ها در این محاسبات آورده شده اند.



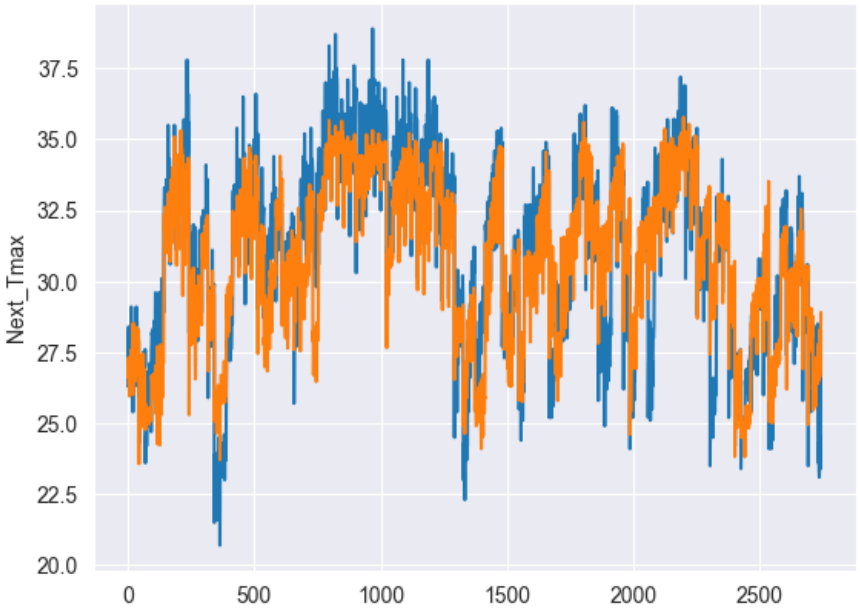
نمودار توزیع داده‌های ویژگی‌ها بصورت دو به دو

B)

ابتدا با یک متود کتابخانه ای این کار را انجام میدهیم

برای حل این مساله از gradient boosting regression استفاده شده است این روش یک متود ensemble هست و با ترکیب مدل‌های ساده، استفاده از درخت تصمیم و بهره گیر از regression قدرت زیادی بدست می‌آورد و مسائل غیر خطی را نیز می‌تواند محاسبه کند. پیچیدگی زمانی این مدل برابر است با O(KMNT) که در آن K تعداد درخت‌ها، M تعداد راندهای تکرار مساله،N تعداد داده‌ها، T پیچیدگی زمانی آموزش درخت‌ها (باتوجه به اینکه باید عمق کمی‌داشته باشند عدد ثابت فرض شده است اما می‌توان این مورد را (h log(h)) نیز فرض کرد که h عمق در خت می‌باشد.

```
params = {
    "n_estimators": 500,
    "max_depth": 5,
    "min_samples_split": 5,
    "learning_rate": 0.03,
    "loss": "squared_error",
}
```

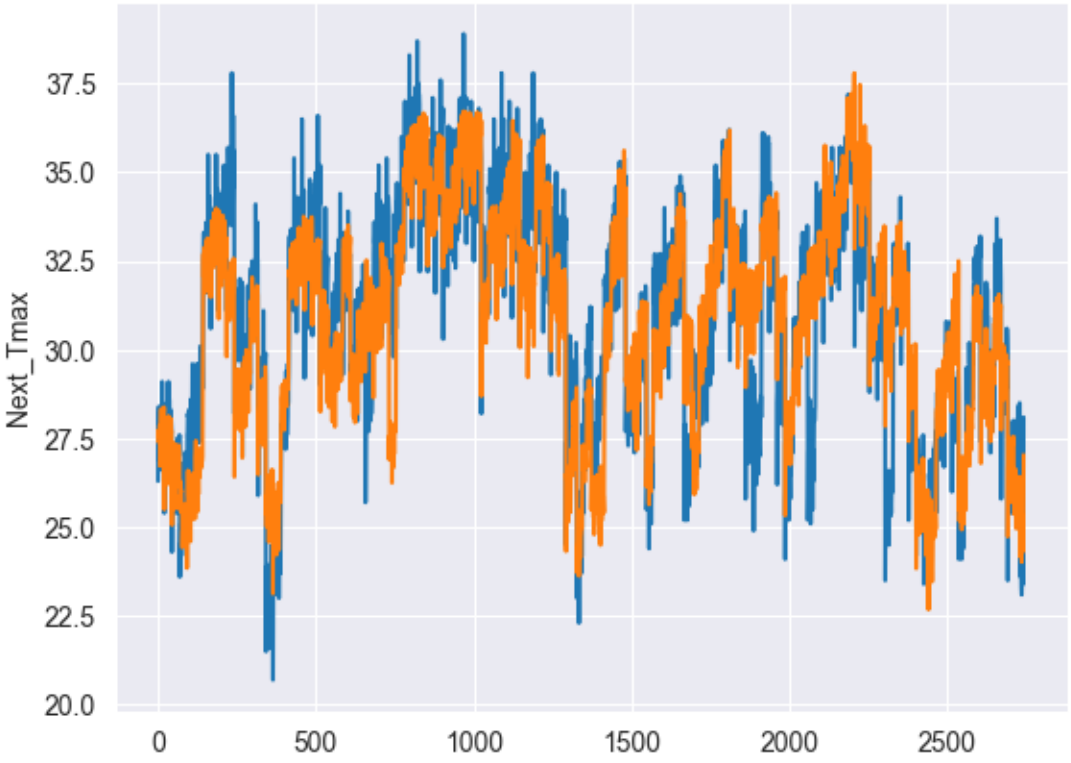



```
The mean squared error (MSE) on test set: 2.6445061843030113
The sum squared error (SSE) on test set: 7251.235957358857
```

سپس با استفاده از روش رگرسیون پیاده سازی شده خودمان جلو می‌رویم که نتایج به شکل زیر هستند

پارامترهای آموزش به شکل زیر می‌باشند

```
learning_rate=0.0001, num_iterations=10000
```




```
The mean squared error (MSE) on test set: 2.600996351389O3
The sum squared error (SSE) on test set: 7131.93199550872
```

در کمال تعجب روش ما بهتر از روش gradient boosting که توسط کتابخانه sikit learn ساخته شده است عملکرد.(البته به مقدار جزئی) پیچیدگی زمانی این مدل O(N * K) می‌باشد که در آن N تعداد داده‌ها K تعداد epoch یا iterationهای ما می‌باشد.

C)

در این بخش با استفاده از می‌خواهیم مدل regression که در بخش قبلی نوشته شده است را توسعه دهیم. برای اینکار یک راه حل ابتدایی و ساده این است که یک مدل بزرگتر داشته باشیم که k تا خروجی متفاوت داشته باشد و k مدل regression را آموزش دهد در این حالت همان regression قبلی را k بار اجرا می‌کنیم. اما راه حل بهتر این است که ماتریس خروجی را k بعدی فرض کرده و ماتریس وزن را نیز k * (features+1) بگیریم در اینجا k تعداد خروجی ما می‌باشد و مقدار 1+ میزان بایاس هر یک از خروجی‌ها می‌باشد. به این شکل regression برای همه خروجی‌ها بصورت ماتریسی محاسبه می‌گردد. نکته حائز اهمیت در این جا این است که بایاس برای مدل‌های متفاوت گرفته شده است در صورتی که این عمل انجام نشود و یک بایاس برای خروجی‌های متفاوت در نظر گرفته شود ممکن است مدل در یکی از پیش بینی‌ها خوب عمل کند و در دیگری خراب باشد یا بالعکس.

D)

در ادامه به ترتیب نتایج هردو فیچر باهم ، Tmax به تنهایی و Tmin به تنهایی را مشاهده می‌کنید.

پارامترهای مدل استفاده شده بصورت زیر می‌باشند

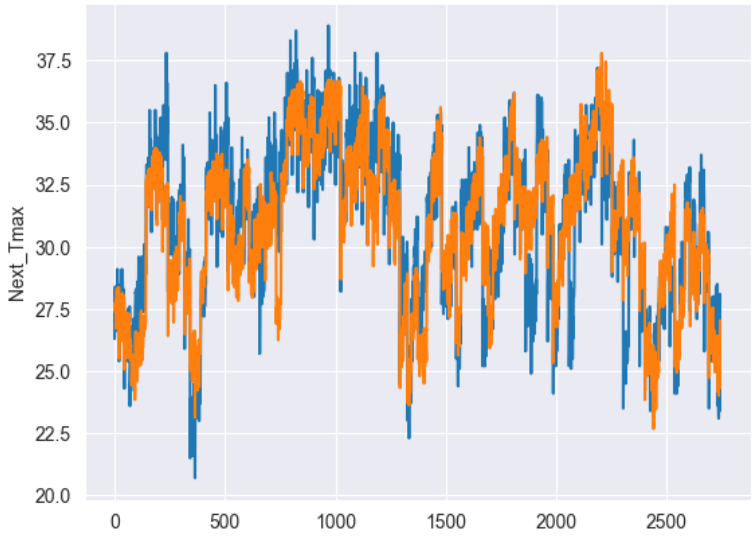
```
learning_rate=0.0001, iterations=10000
```

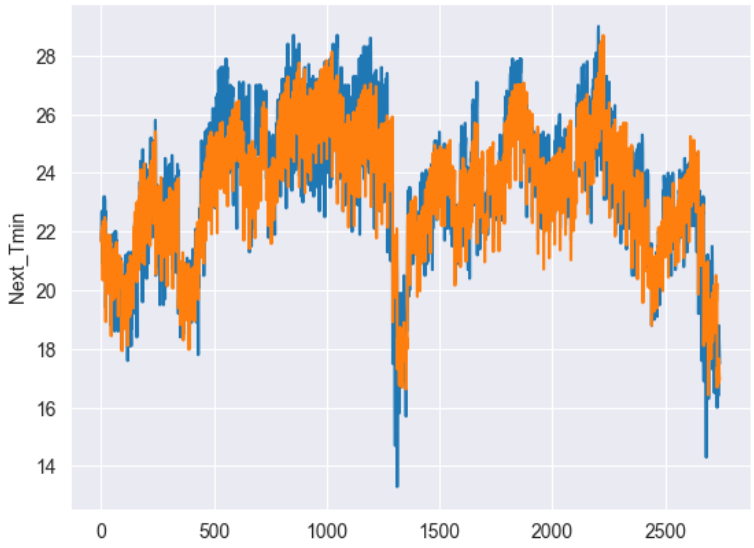
```
The mean squared error (MSE) on test set: 1.7801855500574368
The sum squared error (SSE) on test set: 4881.268778257492
```

```
The mean squared error (MSE) on test set for T_max: 2.600638085344277
The mean squared error (SSE) on test set for T_max: 7130.949630014007
```

```
The mean squared error (MSE) on test set for T_min: 0.9597500268157335
The mean squared error (SSE) on test set for T_min: 2631.6345735287414
```

** در همه تصاویر بخش آبی داده اصلی است ونارنجی مقدار پیش بینی شده است.





E)

نتایج بدست آمده برای حالت دوبعدی از نظر میزان SSE بهبود عددی را نشان می‌دهد ولی این عدد را نباید با خطای قبلی مقایسه کرد. در این مساله دو فیچر تخمین زده شده است که فیچر Next_Tmax به نسبت خطای بیشتری در آن وجود داشته و در Next_Tmin خطا، خیلی کمتر می‌باشد. این شرایط منجر به این می‌شود که وقتی به این دو فیچر به طور همزمان نگاه می‌کنیم حس می‌کنیم عملکرد دوتایی بهبود یافته است اما این چنین نیست و عملکرد تقریبا مشابه بوده است و کاهش میزان خطا به دلیل ترکیب دو فیچر می‌باشد. فرق اساسی این روش بهبود سرعت و امکان موازی سازی بیشتر در ضرب‌های ماتریسی است.

F)

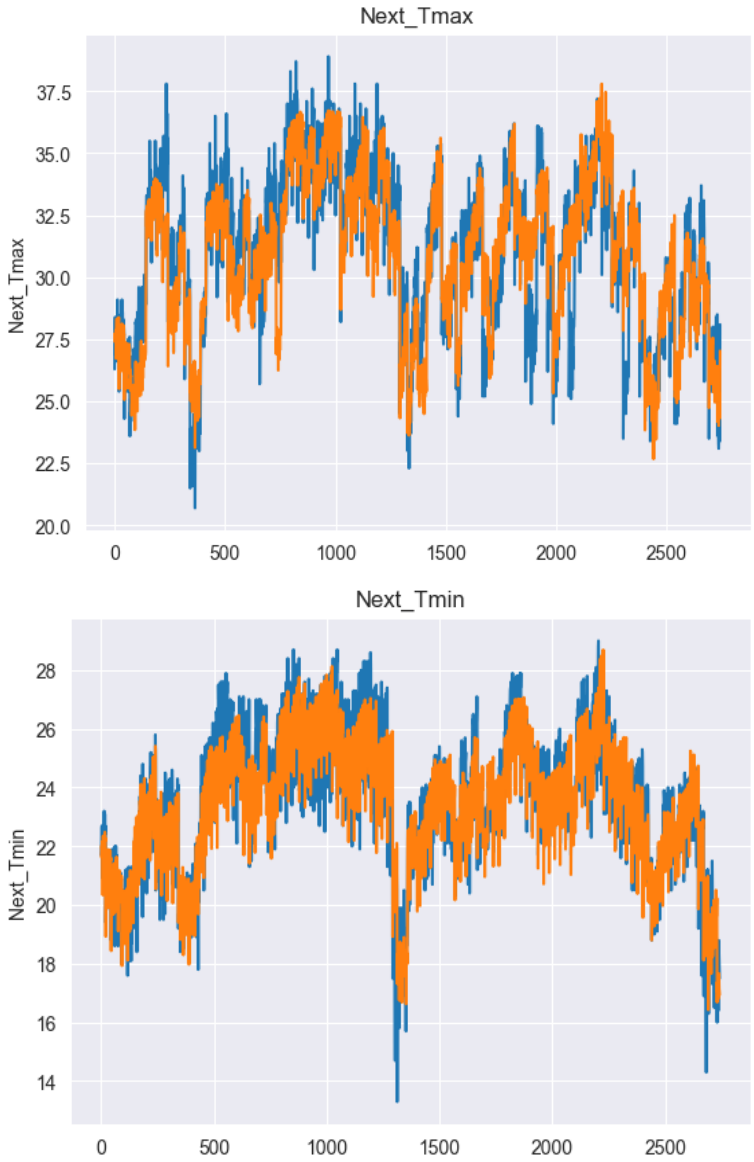
نتایج بدست آمده به شکل زیر می‌باشد.

```
The mean squared error (MSE) on test set: 1.7803738232287858
The sum squared error (SSE) on test set: 4881.785023293331
```

```
The mean squared error (MSE) on test set for T_max: 2.600997619641839
The mean squared error (SSE) on test set for T_max: 7131.935473057923
```

```
The mean squared error (MSE) on test set for T_min: 0.9597500268157335
The mean squared error (SSE) on test set for T_min: 2631.6345735287414
```

نتایج این بخش به ما نشان می‌دهد که مدل‌هایی که با استفاده از نزول در جهت گرادیان در بخش قبل آموزش داده بودیم به درستی همگرا شده اند زیرا نتایجی نزدیک به normal equation از خود نشان داده اند.



g)

در این روش در زمان آموزش صرفا همه داده‌ها ذخیره می‌شوند و پردازش خاصی صورت نمی‌گیرد. زمانی که یک داده تست وارد می‌شود k نزدیک ترین همسایه به آن در داده‌های آموزشی پیدا شده و با میانگین گیری در بین مقادیر آن‌ها مقدار پیش بینی شده تخمین زده می‌شود.

خطای داده بصورت بردار دوبعدی

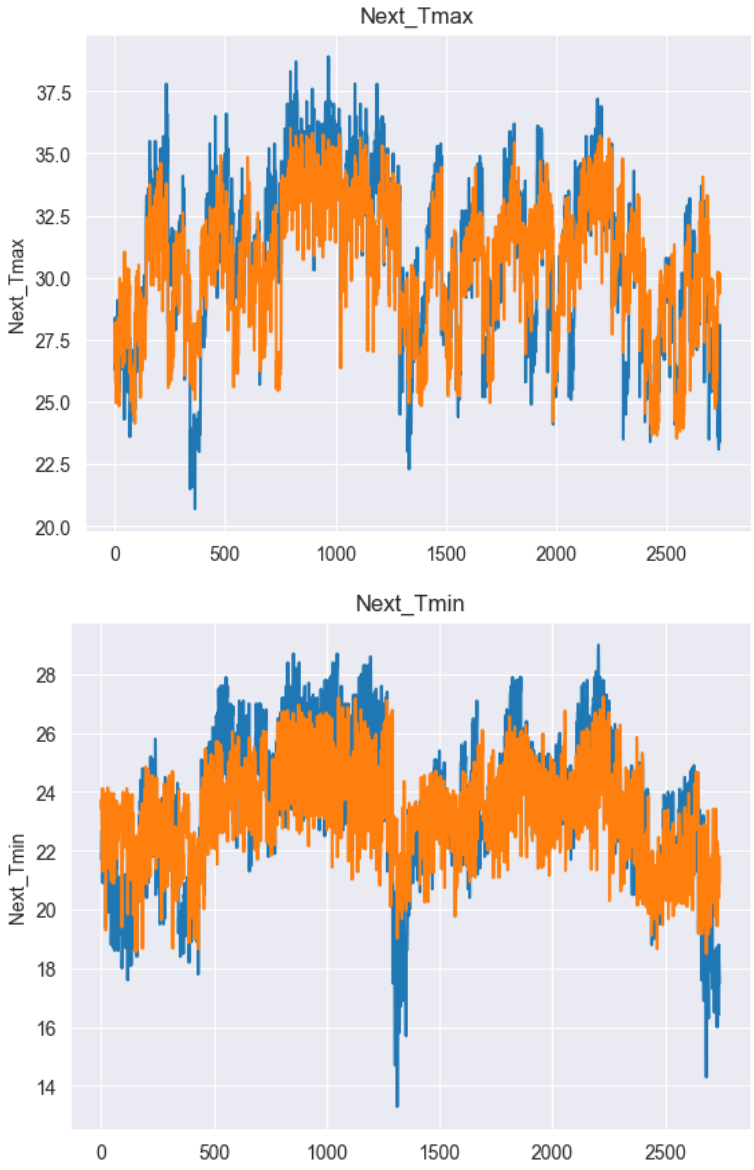
```
The mean squared error (MSE) on test set: 3.2544739606126916
The sum squared error (SSE) on test set: 8923.767600000001
```

خطای Tmax به صورت تنها

```
The mean squared error (MSE) on test set for T_max: 4.037490153172866
The mean squared error (SSE) on test set for T_max: 11070.797999999999
```

خطای Tmin بصورت تنها

```
The mean squared error (MSE) on test set for T_min: 2.4714577680525167
The mean squared error (SSE) on test set for T_min: 6776.7372000000005
```



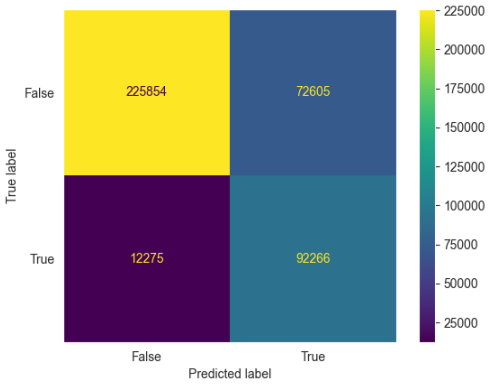
این مدل ضعیف تر از همه مدل‌های موجود عمل کرده است.

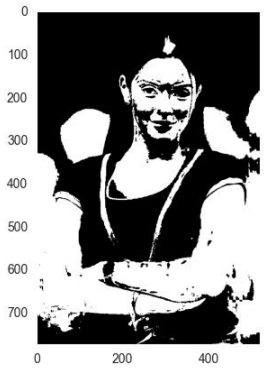
Q2)

A,B)

در این بخش از دو مدل GMM آموزش داده شده است یکی برای داده‌های با لیبل ۱ و یکی برای داده‌های با لیبل ۲ (که در پیاده سازی این لیبل به صفر تغییر کرده است)

دو مدل آموزش داده شده یکی دارای n_components=5 و یکی داری n_components=2 دلیل این تصمیم این است که دسته‌های مربوط به پوست تنوع کمتری نسبت به همه حالات دیگر دارند در نتیجه مدل ساده تری برای تشخیص پوست نیاز است.





Accuracy های بدست آمده برای هر یک از عکس‌ها در زیر آمده‌است

Accuracy 920480_f520: 0.8022307692307692

Accuracy 0520962400: 0.5698883047028324

Accuracy chenhao0017me9: 0.9677547325102881

Accuracy f_family: 0.740011768167108

Accuracy family_bible_study: 0.8614071856287425

Accuracy Family_Bryce: 0.8042354166666666

Accuracy FamilyPhoto07: 0.97002002002002

Accuracy friends: 0.8222978723404255

Accuracy m(01-32)_gr: 0.9074765625

Accuracy RoundsFamily: 0.5577057831221074

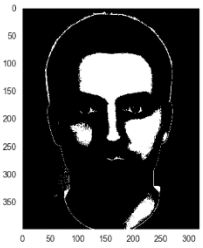
Accuracy vick-family: 0.8115245478036176

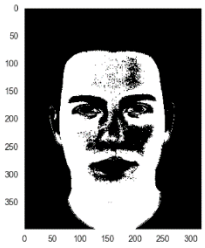
** با توجه به اینکه آوردن همه عکس‌ها در گزارش مقدور نبود confusion ماتریس‌ها و خروجی‌های بدست آمده در بخش report در یک فولد images قرار داده شده است که داخل هر بخش عکس‌های مرتبط با هر مدل قرار داده شده است.

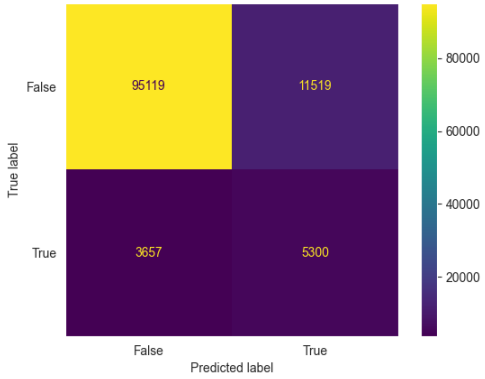
C)

در این بخش از GNB برای تشخیص پوست استفاده شده است. نتیاج بدست آمده بهتر از بخش‌های قبلی بوده اند برای مثال می‌توانیم به عکس‌های زیر اشاره کنیم.

عکس سمت چپ در بالا مربوط به GMM و عکس سمت راست مربوط به GNB می‌باشد. عملکرد بهتر GNB کاملا مشهود می‌باشد. البته در تصاویری که یک عامل بزرگ در حال بررسی است GMM بهتر عمل کرده است مثل تصویر زیر(سمت راست GMM و سمت چپ GNB می‌باشد) اما در کل عملکرد GNB رضایت بخش تر بوده است.







Confustion_matrix(Round family)

** همه تصاویر ساخته شده و ماترسی‌های در هم ریختگی در بخش Report فولدر Q2 بخش C قرار داده شده است.

میزان accuracy های گزارش شده :

Accuracy 920480_f520: 0.7945161290322581

Accuracy 0520962400: 0.6717103190948077

Accuracy chenhao0017me9: 0.9443275720164609

Accuracy f_family: 0.7981685789938218

Accuracy family_bible_study: 0.5442455089820359

Accuracy Family_Bryce: 0.8453458333333334

Accuracy FamilyPhoto07: 0.8255355355355355

Accuracy friends: 0.5549957446808511

Accuracy large_Chapman-family: 0.8918629747827852

Accuracy m(01-32)_gr: 0.72053125

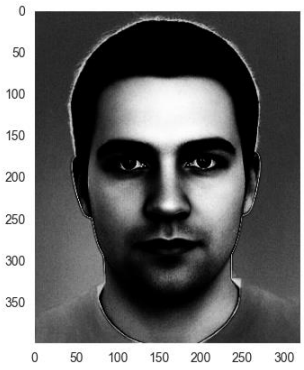
Accuracy RoundsFamily: 0.8687140447251178

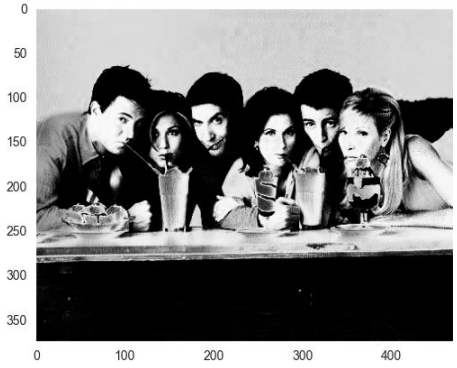
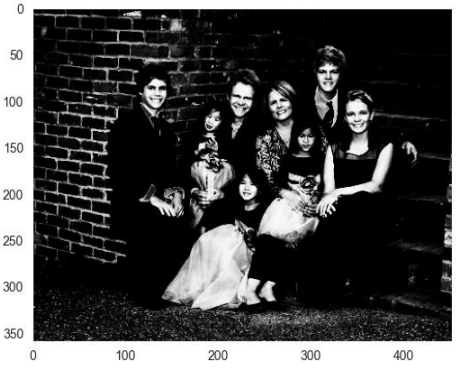
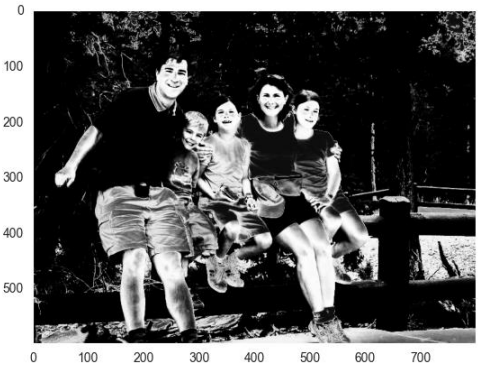
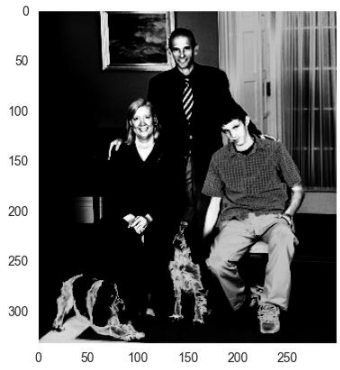
Accuracy vick-family: 0.6759173126614987

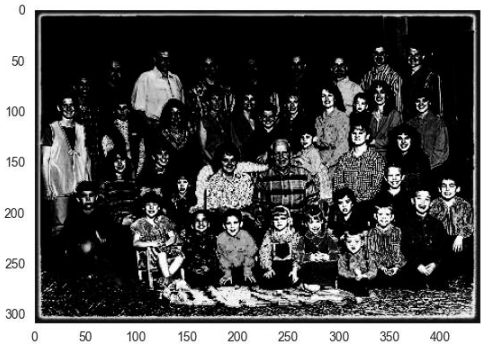
Accuracy در هر دو مدل نزدیک است ولی چیزی که تفاوت می‌کند کیفیت و Recall است که در سری تصاویر این بخش بهتر شده است.

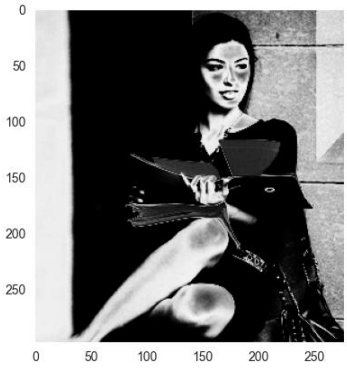
D)

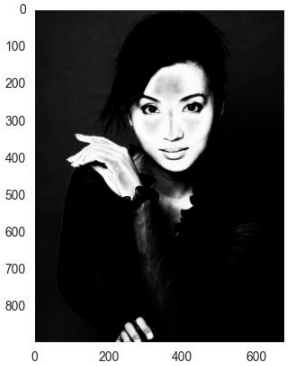
در این بخش از بصورت soft عمل شده و از احتمال assign شدن هر دسته به دسته اصل استفاده شده است.

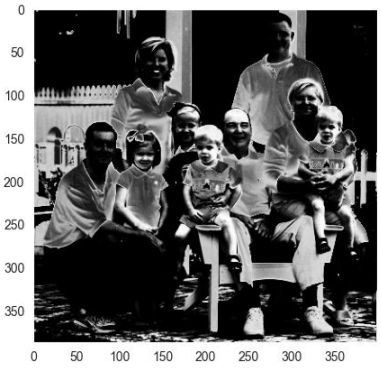


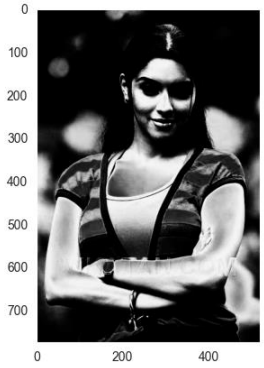















E)

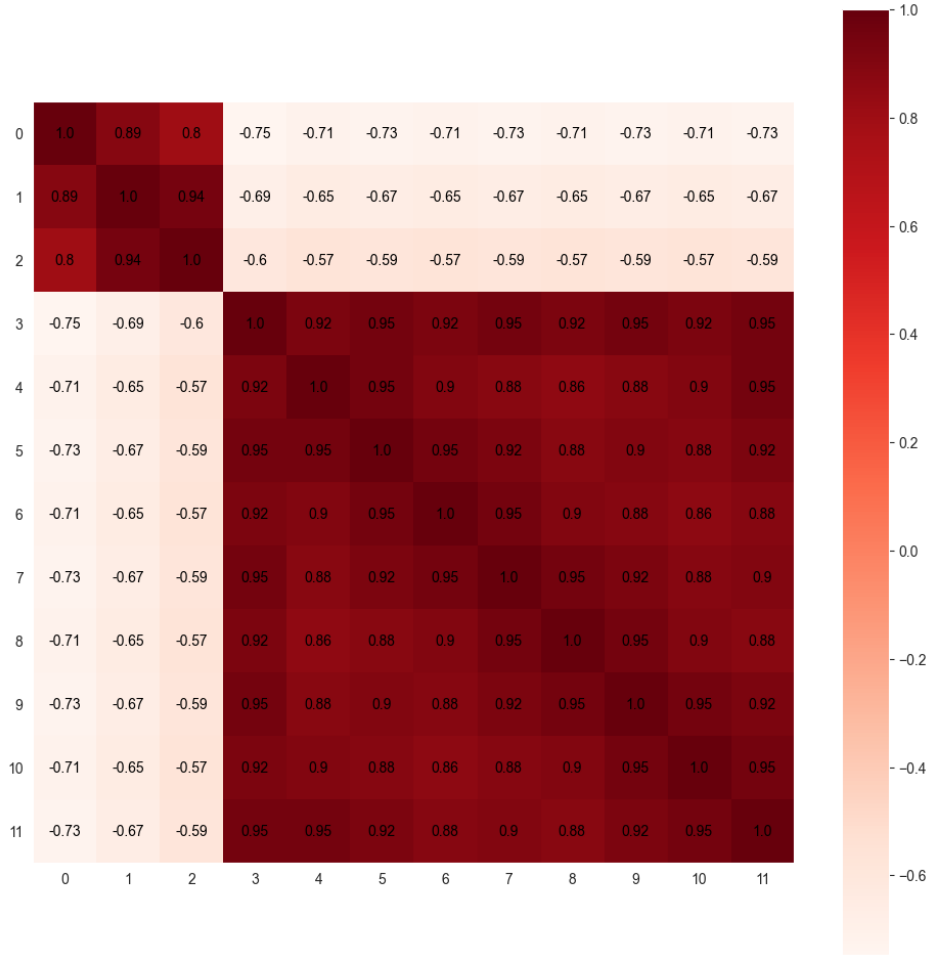
برای اینکه این بخش با سهولت بیشتری انجام شود پس از محاسبه احتمال هر یک از سلول‌ها به آن یک padding داده شده است که مقدار صفر دارد و در نتیجه نیازی به چک کردن ایندکس‌ها بصورت دستی نمی‌باشد. پدینگ داده شده مقدار دو می‌باشد تا بتوان حالت‌های مختلف بیشتری را تست نمود.

همچنین با دوحالت predict_prob و predict_log_prob مراحل بخش بعدی را تست کردیم اما نتیجه predict_prob در نهایت بهتر بود.

F)

نمودار بدست آمده حاکی از آن است که میزان احتمال skin بودن یک نقطه وابسته بسیار زیادی به نقاط همسایگی خودش دارد. لازم به ذکر است در محاسبات مقادیر RGB نرمالایز شده اند تا حساسیت correlation coefficient نسبت به اسکیل بر طرف شود.

باتوجه به جدول می‌توان به این موضوع اشاره کرد که وابستگی نقاط همسایه به نقطه مرکزیی به شکل ویژه ای بیشتر است و دلیل آن هم این است که فاصله همه همسایه‌ها تا مرکز ۱ است در صورتی که با بقیه نقاط فاصله بیشتری دارند و در نتیجه وابستگی کم تری به یک دیگر خواهند داشت.(این موضوع در سطر ۳ و ستون ۳ که رابطه مرکز با سایرین را نشان می‌دهد مشهود است)



G)

عملکرد این روش برای من آنقدر مناسب نیست ولی در تصاویری که صورت‌ها و تصاویر صورت و پوست کوچک هستند خوب عمل کرده است.

میزان accuracy های بدست آمده:

Accuracy 920480_f520: 0.779498759305211

Accuracy 0520962400: 0.7357352264192086

Accuracy chenhao0017me9: 0.8920954732510288

Accuracy f_family: 0.9121285672256546

Accuracy family_bible_study: 0.8590419161676647

Accuracy Family_Bryce: 0.911875

Accuracy FamilyPhoto07: 0.8807507507507507

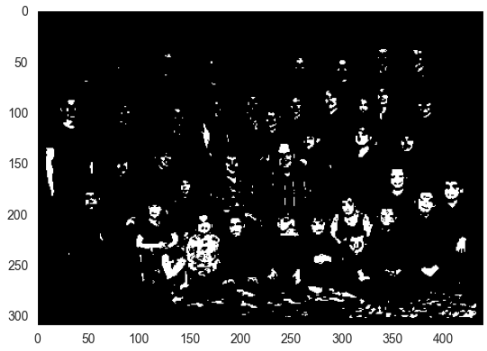
Accuracy friends: 0.912709219858156

Accuracy large_Chapman-family: 0.9277733709654609

Accuracy m(01-32)_gr: 0.71609375

Accuracy RoundsFamily: 0.9047882693888144

Accuracy vick-family: 0.8591666666666666



چند مورد دیگر از تصاویرایجاد شده است.

** مابقی تصاویر در بخش report فولدر images در بخش G قرار داده شده است.

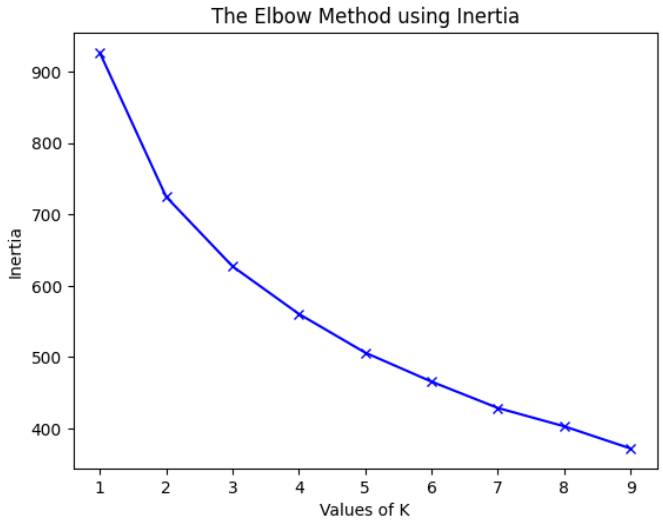
Q3)

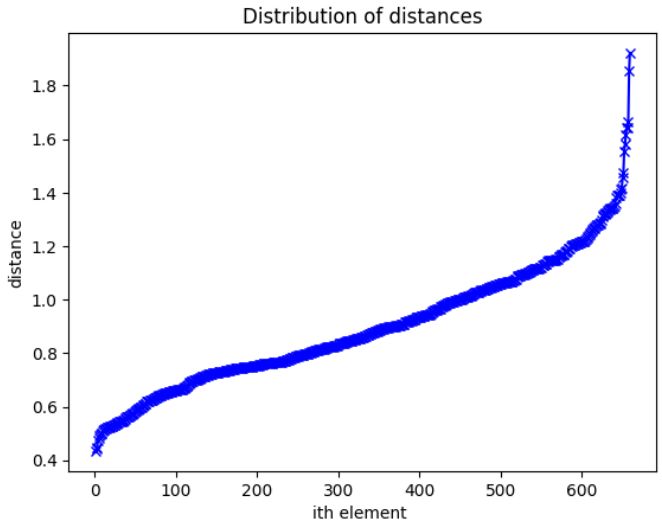
A)

در این بخش ابتدا داده‌های مساله با استفاده از کتابخانه pandas خوانده شده است(در این داده ? به نشانه null فرض گردیده شده است) سطرهایی که داده‌های null داشته اند حذف شده اند. برای اینکه تعداد حذف‌ها کم شود برای سه ستون مقادیر null با میانگین پرشد که تاثیری چندانی در تعداد حذفی‌ها نداشت که این به این معنی است که سطرهای حذف شده تعداد زیادی null داشته اند و حذف آن‌ها کار درستی بوده است.

داده‌هایی که categorical هستند به داده‌های عددی تبدیل شده اند و سپس همه ستون‌ها در بازه ۰ تا ۱ نرمال شده اند.

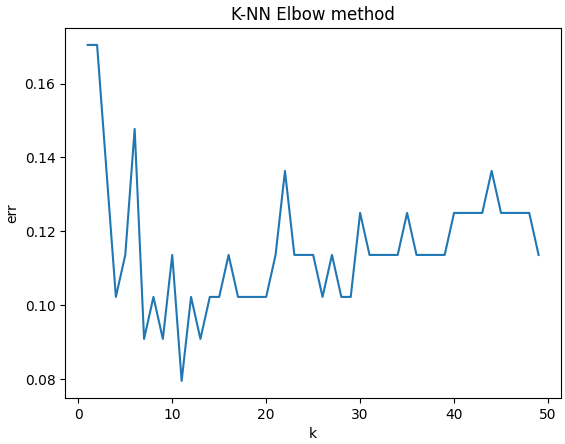
برای حذف داده‌های پرت ابتدا kهای مختلف برای kmeans را تست می‌کنیم. k مورد نظر در این جا ۴ فرض شده است سپس نمودار فاصله تا میانگین برای نقاط را بصورت sort شده ترسیم می‌کنیم و نقاطی که دور تر از 1.2 تا مبدا فاصله دارند را حذف می‌کنیم.





B)

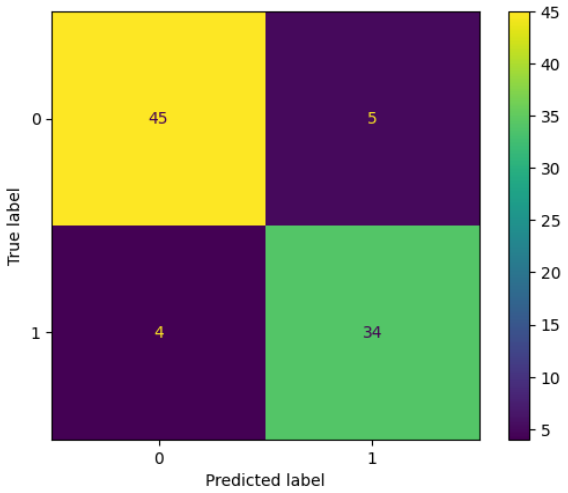
برای استفاده از elbow در این مساله err را 1- accuracy تعریف کرده ایم و برای kهای مختلف نتیجه را رسم کردیم. با توجه به شکل زیر k =7 را می‌توانیم جایی در نظر بگیریم که به فضای نسبتا مسطحی منجر می‌شود و با این مقدار مساله را ادامه دادیم.




```
Time training => 0.0050160884857177734 seconds
```

```
Time for prediction one data point => 0.002996206283569336 seconds
```

	precision	recall	f1-score	support
0	0.92	0.90	0.91	50
1	0.87	0.89	0.88	38
accuracy			0.90	88
macro avg	0.90	0.90	0.90	88
weighted avg	0.90	0.90	0.90	88

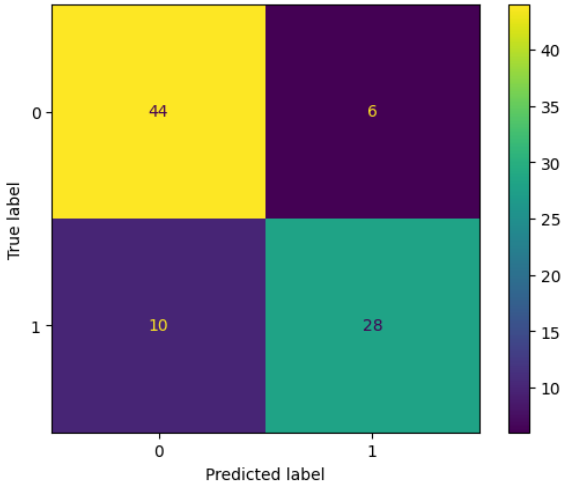


C) Decision Tree

```
Time training => 0.005020618438720703 seconds
```

```
Time for prediction one data point => 0.0019986629486083984 seconds
```

	precision	recall	f1-score	support
0	0.81	0.88	0.85	50
1	0.82	0.74	0.78	38
accuracy			0.82	88
macro avg	0.82	0.81	0.81	88
weighted avg	0.82	0.82	0.82	88



D) GridSearch _ RandomForestClassifier

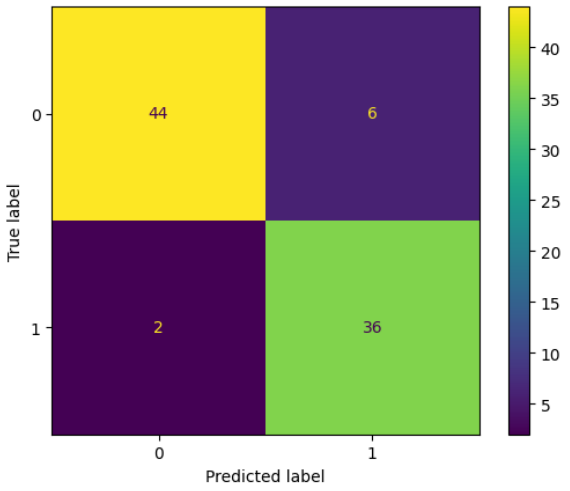
زمان آموز بیش از ۵ ثانیه بوده است.

```
Time training => 5.262007474899292 seconds
```

```
Time for prediction one data point => 0.0029954910278320312 seconds
```

این عدد به نسبت مدت زمان زیادی برای دیتاست کوچک ما می‌باشد اما در زمانی که به دنبال یافتن بهترین جواب ممکن هستیم بهترین راه حل برای یافتن hyper parameterها می‌باشد. البته برای بهبود سرعت می‌توانیم سعی کنیم فقط برای فیچرهایی که اهمیت بیشتری دارند حالات متفاوت در نظر بگیریم و حالات با اهمیت کمتر را جدا گانه تست کنیم تا تعداد جایگشت‌های کمتری ایجاد شود.

	precision	recall	f1-score	support
0	0.96	0.88	0.92	50
1	0.86	0.95	0.90	38
accuracy			0.91	88
macro avg	0.91	0.91	0.91	88
weighted avg	0.91	0.91	0.91	88

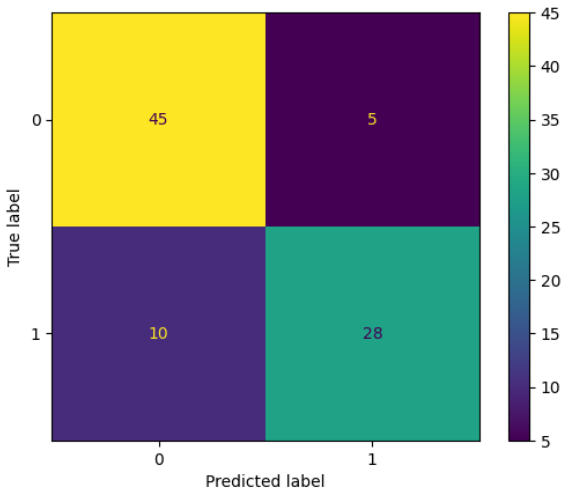


E) GuassianNB

```
Time training => 0.004000186920166016 seconds
```

```
Time for prediction one data point => 0.0019910335540771484 seconds
```

	precision	recall	f1-score	support
0	0.82	0.90	0.86	50
1	0.85	0.74	0.79	38
accuracy			0.83	88
macro avg	0.83	0.82	0.82	88
weighted avg	0.83	0.83	0.83	88

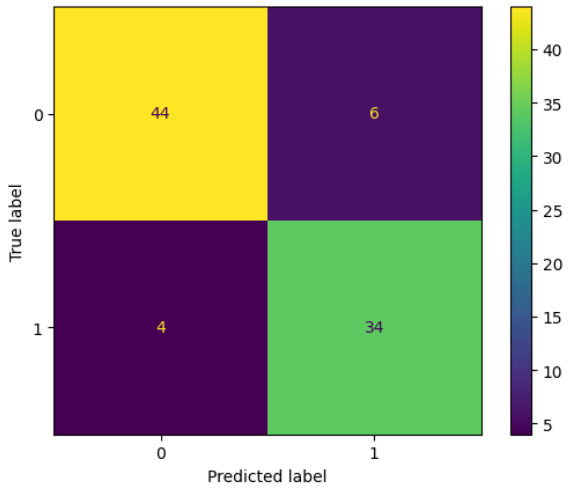


F) LogesticRegression

```
Time training => 0.0059947967529296875 seconds
```

```
Time for prediction one data point => 0.0019943714141845703 seconds
```

	precision	recall	f1-score	support
0	0.92	0.88	0.90	50
1	0.85	0.89	0.87	38
accuracy			0.89	88
macro avg	0.88	0.89	0.88	88
weighted avg	0.89	0.89	0.89	88

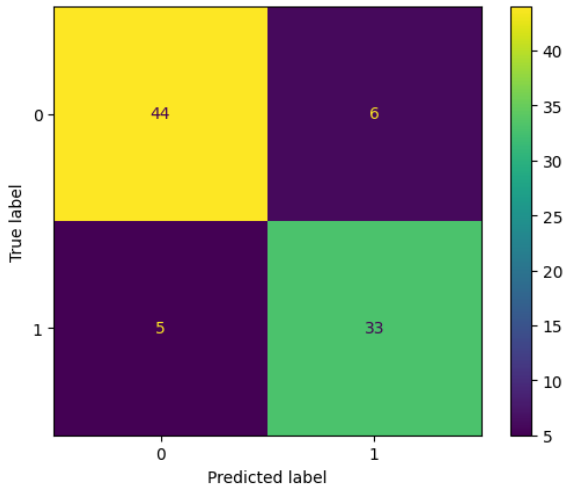


G) SVM

```
Time training => 0.00699925422668457 seconds
```

```
Time for prediction one data point => 0.0010018348693847656 seconds
```

	precision	recall	f1-score	support
0	0.90	0.88	0.89	50
1	0.85	0.87	0.86	38
accuracy			0.88	88
macro avg	0.87	0.87	0.87	88
weighted avg	0.88	0.88	0.88	88

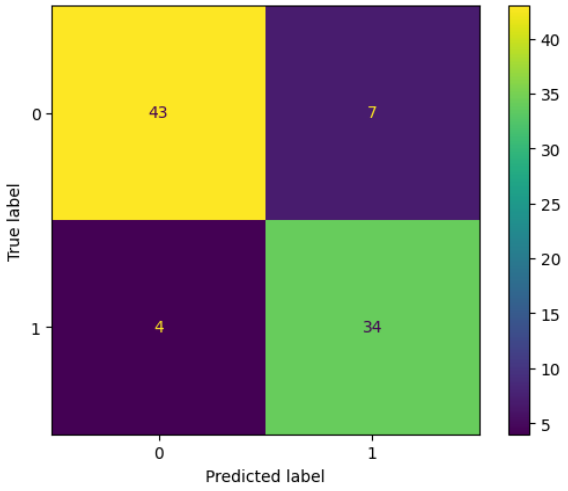


H) AdaBoostClassifier

```
Time training => 0.09200692176818848 seconds
```

```
Time for prediction one data point => 0.007009029388427734 seconds
```

	precision	recall	f1-score	support
0	0.91	0.86	0.89	50
1	0.83	0.89	0.86	38
accuracy			0.88	88
macro avg	0.87	0.88	0.87	88
weighted avg	0.88	0.88	0.88	88



i)

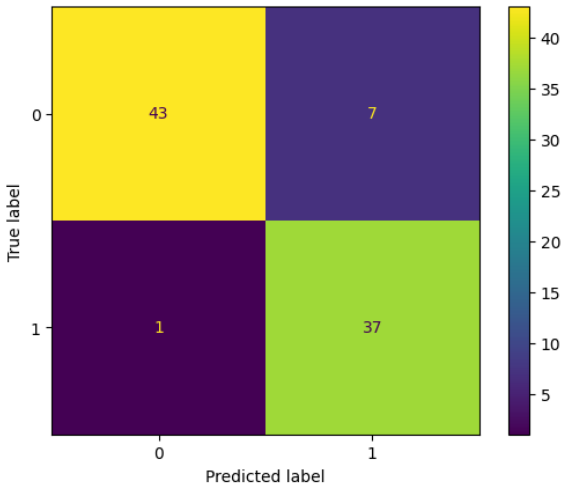
در k-means پیاده سازی شده ابتدا الگوریتم مورد نظر اجرا شده و مراکز دسته برای داده‌های آموزشی بدست می‌آید. درگام بعدی باید لیبل کلاسترها را انتخاب کنیم. اتفاقی که در پیاده سازی افتاده است این است که ابتدا به دو کلاستر بصورت رندوم ۰ و۱ اختصاص داده می‌شود درصورتی که تعداد اشتباهات بیشتر از تصمیم‌های درست باشد جای ۰ و ۱ لیبل‌ها را عوض می‌کنیم. این کار معادل رای گیری می‌باشد.

** در بخش پایانی این سوال k-means باقابلیت رای گیری بین چند دسته نیز نوشته شده است که در همان قسمت توضیحات تکمیلی آورده شده است.

```
Time training => 0.0249929428810058594 seconds
```

```
Time for prediction one data point => 0.005000114440917969 seconds
```

	precision	recall	f1-score	support
0	0.98	0.86	0.91	50
1	0.84	0.97	0.90	38
accuracy			0.91	88
macro avg	0.91	0.92	0.91	88
weighted avg	0.92	0.91	0.91	88



j)

	FN	FP	accuracy	Precision (For +)	F1_score (For +)	Recall (For +)	Train time	Predict time
K-NN	4	5	0.90	0.87	0.88	0.89	0.00501	0.00299
Decision Tree	10	6	0.82	0.82	0.78	0.74	0.00502	0.0019
Random Forest	2	6	0.91	0.86	0.90	0.95	5.2620	0.0029
GNB	10	5	0.83	0.85	0.79	0.74	0.0040	0.0019
Logistic Regression	4	6	0.89	0.85	0.87	0.89	0.00599	0.0019
SVM	5	5	0.88	0.85	0.86	0.87	0.0069	0.0010
AdaBoost	12	20	0.88	0.83	0.86	0.89	0.092	0.0070
K-Means	1	7	0.91	0.84	0.90	0.97	0.0249	0.0050

در ستون‌های FN و FP از آنجا که نرخ در خواست نشده است تعداد آن‌ها وارد گردیده است.

در بین مدل‌ها Random Forest و K-Means بیشترین accuracy را داشته اند اما Random Forest استفاده شده چون از grid search بهره می‌برد مدت آموزش بسیار زیادی دارد نسبت به بقیه، همچنین میزان Recall در این مدل برروی برچسب + بیشتر از سایرین بوده است که در صورتی که هدف آموزش یافتن لیبل‌های + باشد می‌توان به عنوان مزیت برای این روش‌ها به حساب بیاید.

kNN نیز در این مساله عمل کرد خوبی داشته است(جایگاه سوم در accuracy با 0.01 اختلاف با نفر اول) اما با توجه به اینکه تقریبا کار خاصی در بخش آموزش نمی‌کند و فقط ذخیره سازی را انجام می‌دهد در بخش تست بار محاسباتی بیشتری را تحمل می‌کند برای همین در بخش پیش بینی جزو زمان برترین روش‌ها به حساب میاد (جایگاه دوم). همچنین KNN در بین مدل‌ها بیشترین precision را داشته است.

بدترین عملکرد نیز به طور مشترک به GNB و DecisionTree اختصاص داشته است ، البته باید این نکته را متذکر شد این عملکرد بر روی این دیتاست بوده است و دلیل بر برتری یک مدل بر دیگری در تمام شرایط ندارد بلکه به توزیع داده‌ها و یا حتی با توجه به نزدیکی نتایج به عملگر تصادفی که داده‌های آموزش و تست را جدا کرده است بستگی دارد.

** این نتایج به نحوه تشخیص داده پرت و میزان آن نیز بستگی دارد.

K)

برای بهبود در بهترین مدل بهبود در مدل kmeans را انتخاب کردم و ایده من برای بهبود این روش این بوده است که از kهای بیشتر برای این کار استفاده کنیم مثلا با k=5 یا بیشتر بعد از اینکه centroidها مشخص شده اند با استفاده از مدل KNN که در بخش‌های قبلی آموزش داده ایم کلاس خوشه‌ها را مشخص می‌کنیم. سپس برای داده‌های ورودی نزدیک ترین کلاستر را یافته و لیبل اختصاص داده شده به آن را به عنوان پاسخ اعلام می‌کنیم.

این روش در حالات پیچیده قدرت زیادی به ما می‌دهد اما در اینجا به دلیل ساده بودن داده‌ها نتیجه ای مشابه حالت kmeans دوتایی برای ما ایجاد کرد.

```
Time training => 0.027992248535156 25 seconds
```

```
Time for prediction one data point => 0.0009996891021728516 seconds
```

	precision	recall	f1-score	support
0	0.98	0.86	0.91	50
1	0.84	0.97	0.90	38
accuracy			0.91	88
macro avg	0.91	0.92	0.91	88
weighted avg	0.92	0.91	0.91	88