به نام خدا

دانشگاه صنعتی امیرکبیر (پلی تکنیک تهران)

دانشکده مهندسی کامپیوتر

یادگیری ماشین

تکلیف سوم

استاد درس: دکتر ناظرفرد

امیرحسین کاشانی

۴۰۰۱۳۱۰۷۱

amkkashani@gmail.com

نیم سال اول ۱۴۰۱-۱۴۰۲

فهرست

Q1)

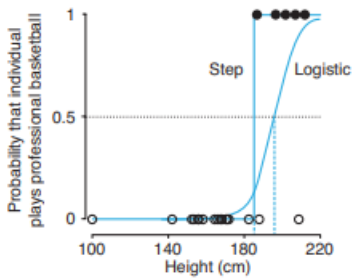
a)

در انتخاب threshold ما به دنبال بهبود classification خود هستیم. در صورتی که توزیع داده بالانس باشد به راحتی می توانیم از همان threshold ۰.۵ استفاده کنیم یا اینکه با سعی خطا بر روی معیار accuracy یک عدد مناسب پیدا کنیم.

اما در صورت unbalance بودن دیتا، مثل تشخیص ایمیل اسپم، تشخیص بیماری و ... توزیع کلاس ها یکسان نیستند. دو روش کلی برای انتخاب threshold وجود دارد که اولی بر روی classifier تمرکز دارد اما دومی توجهش بیشتر به داده می باشد در روش اول به پارامتر های کلی توجه می شود برای مثال accuracy کل مد یا precision مواردی هستند که در روش اول اهمیتی زیادی دارند. مدل اول در مواجهه با دیتایی که ۵ درصد ورودی ها اسپم هستند و ۹۵ درصد سالم هستند به این سمت می رود که با سالم اعلام کردن همه دقت ۹۵ درصدی را تضمین کنم و به پارامتر هایی که روی یک کلاس تعیین می شود توجه نمی کند.

اما روش دوم به این سمت می رود که threshold را به گونه ای تعریف کند که برای زیر کلاس های خاص شرایط بهتری را فراهم کند. برای مثال Recall کلاس اسپم را بهبود دهد و افزایش Recall در این مورد از کاهش دقت مهم تر است.

b



منبع

H. Zhang, Z. Li, H. Shahriar, L. Tao, P. Bhattacharya, and Y. Qian, ‘Improving prediction accuracy for logistic regression on imbalanced datasets’, in *2019 IEEE 43rd Annual Computer Software and Applications Conference (COMPSAC)*, 2019, vol. 1, pp. 918–919.

b)

Multinomial Logistic Regression

در این روش به جای تقسیم مساله به دو کلاس می خواهیم از بین چند کلاس (بیش از ۲) برای داده ورودی انتخاب کنیم. از این رو به جای اینکه از sigmoid استفاده شود از softmax استفاده می کنیم تا مجموع اعداد بدست آمده برابر یک شود و بزرگ ترین مقدار متناظر با لیبل کلاس ها است را به عنوان کلاس برگزیده برای داده ورودی انتخاب می کنیم. برای تعریف تابع loss از cross_entropy استفاده می کنیم که تفاوت میان دو توزیع احتمالاتی را بیان می کند.

$$H(p,q) = -\sum_x p(x) \log q(x).$$

به بیان دیگر این تابع زمانی صفر می شود که دقیقا p و q توزیع یکسانی داشته باشند. از این رو تابع loss را این گونه تعریف می کنیم.

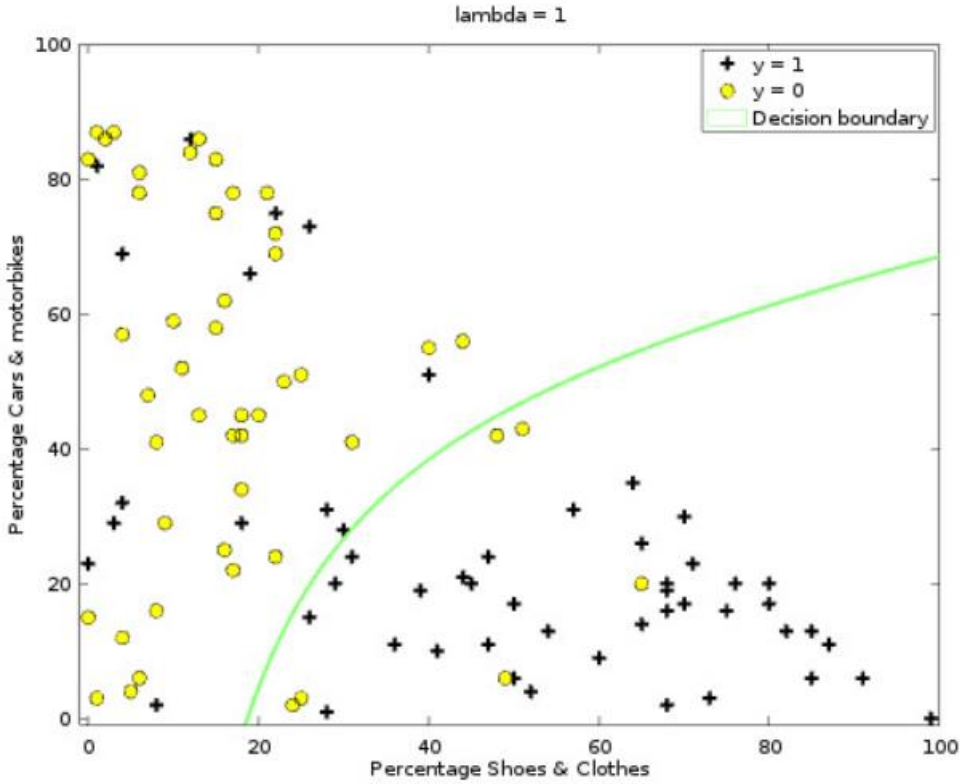
$$\begin{aligned} \text{loss} &= H(y, \hat{y}) \\ &= -\sum_i y_i \log \hat{y}_i \\ &= -\log \hat{y}_c \end{aligned}$$

علت حذف y_i در این معادله این است که اگر صفر باشد کل معادله صفر می شود و اگر یک باشد ضریب خنثی برای ضرب است. با مینیم کردن این تابع به maximum likelihood دست خواهیم یافت. این روش را از آنجا که بر روی تعداد دسته ها محدودیتی ندارد و فقط مقدار تخمین زده شده برای لیبل حقیقی در محاسبه خطا شرکت می کند مستقل از تعداد کلاس می باشد و حتی می تواند در مسائل دسته بندی باینری و چندگانه کاربرد داشته باشد. (از سایت medium)

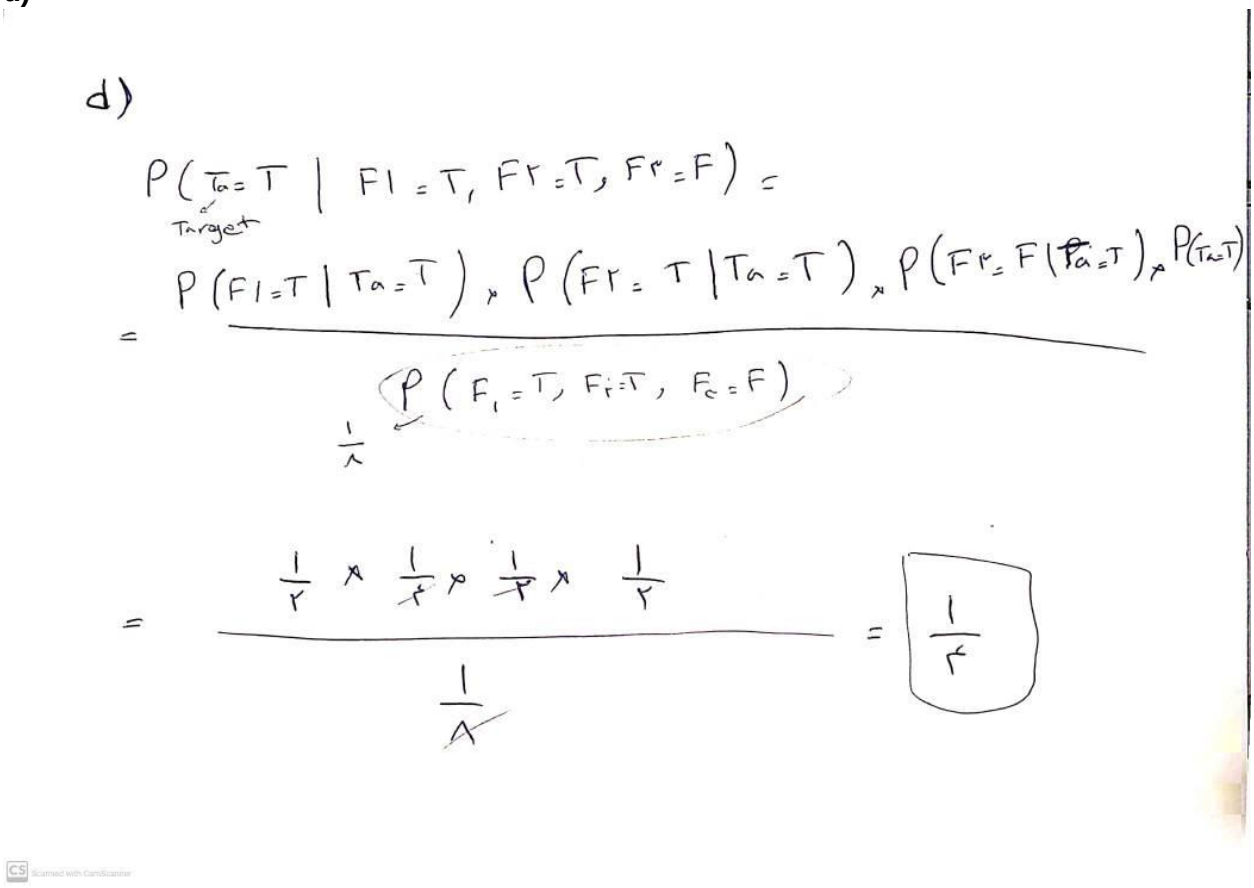
c)

بله

درصورتی که از تابع فعل سازی sigmoid در آموزش logistic regression بهره برده شود می توانیم حالات غیر خطی را نیز پشتیبانی کنیم و مرز هایی مانند شکل زیر ایجاد کنیم


lambda = 1
y = 1
y = 0
Decision boundary
Percentage Cars & motorbikes
Percentage Shoes & Clothes
0
20
40
60
80
100

d)



e)

e.1)

e) $\mu = 1.4$

e.1) $P(ch = No \mid Mar = \text{"Married"}) = \frac{4}{4} = 1$

e.2)

e.2) $f(x) = \frac{1}{\sqrt{2\pi}\sigma} * e^{\frac{-1}{2}\left(\frac{x-\mu}{\sigma}\right)^2}$

$\mu = 90$

$[85, 90, 95] \longrightarrow S = 5$

$S^2 = 25$

$f(x) = \frac{1}{2.5 * 5} * e^{\frac{-1}{2}\left(\frac{2.5}{5}\right)^2} = \frac{e^{-1.8}}{12.5} = 1.2 * 10^{-9}$

e.3)

e.۳)

$$P(X \mid ch = No)$$

$$= P(Refund = No \mid ch = No) \times P(Marital = Married \mid ch = No) \times P(income = 120 \mid ch = No)$$

$$= \frac{4}{7} \times \frac{4}{7} \times 0.0077 = 0.0024$$

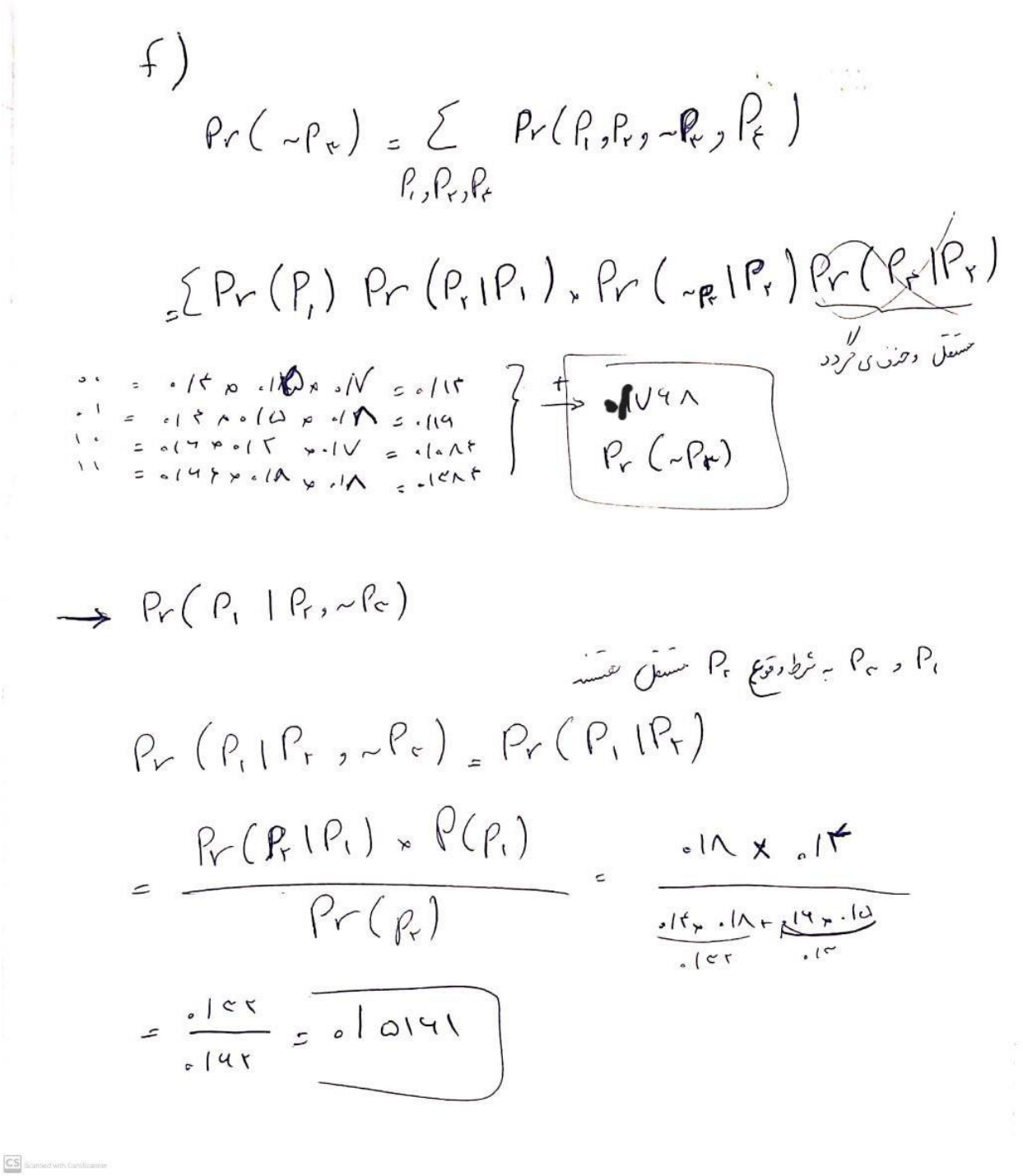
$$P(X \mid ch = Yes) =$$

شرط‌ها به طور مشابه بخش قبل

$$= P(Refund \mid ch = yes) \times P(Marital = Married \mid ch = yes) \times P(income = 120 \mid ch = yes)$$

$$= 1 \times 0 \times 1/2 \times 10^{-9} = 0$$

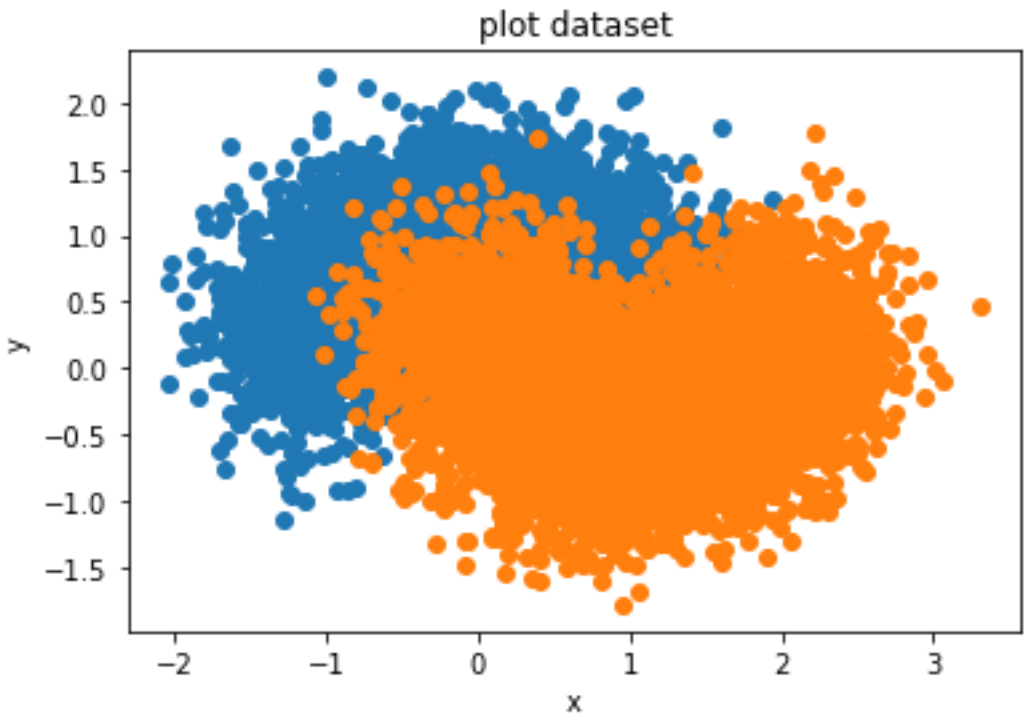
f)



Q2)پ

a)

نتیجه بدست امده برای داده های بدست آمده



b)

```
print(f'X_test :{X_test.shape } - X_train : {X_train.shape}')
```

```
X_test :(2000, 2) - X_train : (8000, 2)
```

c)

grid-search یک روش جست وجو برای hyper parameters می‌باشد که جواب بهینه را می‌یابد. (جواب بهینه مدلی است که بیشترین دقت را پیدا می‌کند) این روش در کل کار خاصی نمی‌کند صرفا همه حالات ممکن برای مدل را اجرا می‌کند و با یکدیگر مقایسه می‌کند.

در ادامه نتیجه نهایی اجرای کد را مشاهده می‌کنید

```
tree_class = DecisionTreeClassifier(random_state=1024)
grid_search = GridSearchCV(estimator=tree_class, param_grid=param_grid, cv=5)  # cross validation
grid_search.fit(X_train, y_train)
```

```
GridSearchCV(cv=5, estimator=DecisionTreeClassifier(random_state=1024),
             param_grid={'max_depth': [4, 6, 8, 9, 10, 15],
                         'max_leaf_nodes': [8, 16, 24, 32, 64, 128, 256, 512]})
```

```
final_model = grid_search.best_estimator_
final_model
```

```
DecisionTreeClassifier(max_depth=8, max_leaf_nodes=16, random_state=1024)
```

d)

کانفیوژن ماتریکس

	0	1
0	839	192
1	105	864

صحت

```
accuracy_score(y_test, pred)
```

```
0.8515
```

e)

این فانکشن بصورت انتخاب رندوم از داده های ورودی بصورت دستی نوشته شده است و به تعداد ۱۰۰۰ دسته ، ۱۰۰ تایی از مجموعه داده های مورد نظر انتخاب می‌کند.

f)

در این بخش از پارامتر های بهترین درخت بدست آمده استفاده می‌کنیم و با کلون کردن آن مدل های کوچکتر را آموزش می‌دهیم.

در زیر بخشی از صحت های بدست آمده است

```
accuracy_score is 0.813
accuracy_score is 0.8075
accuracy_score is 0.8525
accuracy_score is 0.8125
accuracy_score is 0.8455
accuracy_score is 0.8285
accuracy_score is 0.8485
accuracy_score is 0.853
accuracy_score is 0.785
accuracy_score is 0.85
accuracy_score is 0.8295
```

g,h)

در این بخش با استفاده از رای گیری در بین ۱۰۰۰ درخت موجود نظر نهایی این مدل را بدست آوردیم نتیجه نهایی مقدار ناچیزی از بهترین درخت ایجاد شـده در بخش d بهتر اسـت. نتیجه میزان دقت درخت برابر ۸۵/۸ و تک درخت آموزش داده شـده در بخش d برابر ۸۵/۱ می‌باشد که بهبود نسبتا خوبی به شمار می‌آید.

Q3)

a)

در این بخش درخت مورد نظر به صورت بازگشتی تعریف شده است به اینصوت که هر نود یک درخت چپ و یک درخت راست دارد(در صورتی که leaf نباشد) و همچنین اطلاعات داده هایی که در فاز training تا این نود پیش آمده اند را نیز در خود نگه می‌دارد. در این درخت امکان ندارد که فقط یکی از دو نود چپ یا راست وجود داشته باشند زیرا gain این تقسیم صفر می‌شود از این رو یا هردو نود None هستند یا هر دو نود وجود دارند. در بخش آموزش درخت امکان محدود عمق درخت در نظر گرفته شده و همچنین پارامتر purity نیز وجود دارد که می‌گویید که اگر خلوص دسته ای از آن حد بیشتر شد کار متوقف شود از آن جا در اینجا مساله این حالات را از ما نخواسته است عمق برابر تعداد feature ها و معیار purity برابر ۱ در نظر گرفته شده است که این کار معادل خنثی کردن اثر این دو پارامتر در اجرای درخت می‌باشد.

** با توجه به اینکه در این سوال الگوریتم ID3 اجرا می‌شود نیازمند آنیم که داده ها بصورت کتیگوریکال قابلیت دسته بندی داشته باشند از این رو ویژگی سن را با threshold ۵۷ به دو دست متفاوت تقسیم کردیم.

```
accuracy_score : 0.8577586206896551, precision_score : 0.6538461538461539
```

Predicted Actual	NO	YES
NO	17	9
YES	24	182

b)

```
avg accuracy : 0.8509852216748769
avg precision : 0.5470831303343618
```

مسلما نتیجه ی تکرار های متفاوت یکسان نیست زیرا درخت های ما از داده های متفاوتی بوجود می‌آیند و در نتیجه اولیت اجرا مقایسه ها متفاوت است و درخت های متفاوتی تشکلیل خواهد شد.

c)

تکرار بخش a

```
train_size 0.2 => accuracy_score : 0.875, precision_score : 0.28125
train_size 0.45 => accuracy_score : 0.8823529411764706, precision_score : 0.45454545454545453
train_size 0.65 => accuracy_score : 0.8715596330275229, precision_score : 0.38461538461538464
train_size 0.85 => accuracy_score : 0.851063829787234, precision_score : 0.5833333333333334
```

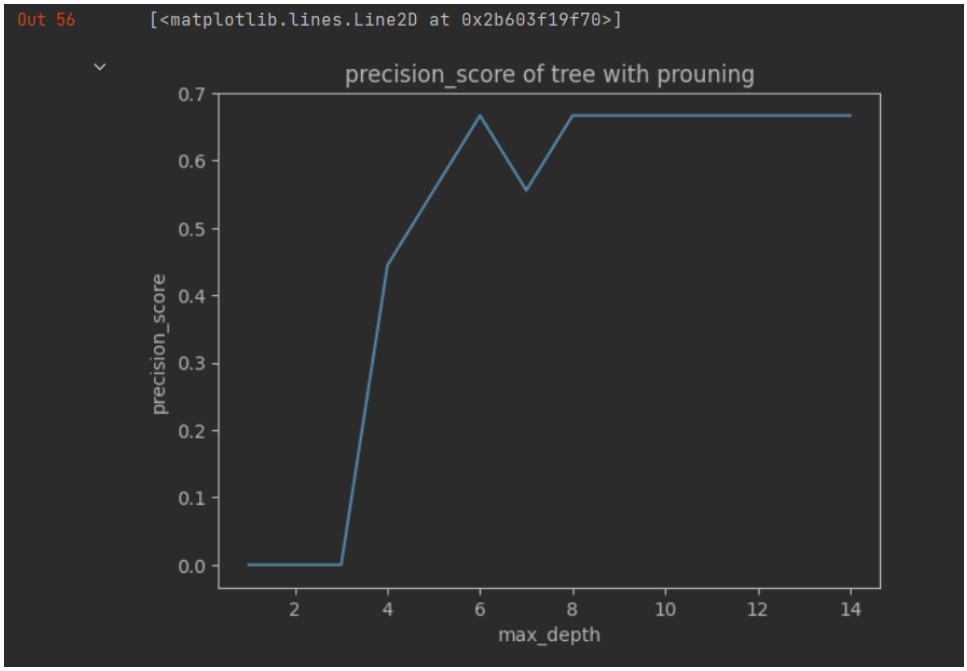
تکرار بخش b

```
train_size : 0.2 => avg accuracy : 0.8605990783410139, avg precision : 0.44534152548991884
train_size : 0.45 => avg accuracy : 0.8722689075630253, avg precision : 0.6210266715381805
train_size : 0.65 => avg accuracy : 0.8951507208387942, avg precision : 0.5605302727151467
train_size : 0.85 => avg accuracy : 0.8996960486322187, avg precision : 0.7054421768707482
```

همان طور که مشـــاهده می‌کنید افزایش داد های آموزش در درخت موجود باعث بهبود عملکرد مخصـــوص در بخش precision می‌باشـد (precision بر روی No فرض شـده اسـت که تعداد کمتری از داده ها را تشـکیل می‌دهد) اما به مرور زمان تاثیر آن روی accuracy کمتر شـده و precision را بهبود می‌دهد و مثلا بعد از مدتی به همگرایی خواهیم رسـید که باتوجه به داده های بیان شده برای تست هنوز نمی‌توان ادعایی بر روی همگرایی آموزش مخصوصا برای precision بیان نمود.

d)

برای عملیات هرس ۲ نوع هرس در ایجاد درخت در نظر گرفته شده است که شامل محدودیت عمق در زمان ساخت و چک کردن میزان خلوص دسته است که در صورتی که یک نود خلوص بیش از آن مقدار را داشته باشد دیگر ادامه به تقسیم شدن نمی‌دهد. که در تست های انجام شده با توجه به اینکه در سوال گفته شده هرس بر روی داده های تست انجام گیری منظور post prune بوده در نتیجه از دو حالت شرط هرس در هنگام سرچ استفاده شده، شرط اول که همان عمق است با این تفاوت که عمق در زمان جست و جو را محدود می‌کند و حالت دوم نیز برای هر نود بررسی می‌کند که آیا میزان بهبود تابع دقت در صورت تجزیه شدن آن نود به دو دسته چه قدر است در صورتی که از مقدار alpha که به تابع داده شده کمتر باشد دیگر جست و جو را ادامه نمی‌دهد و در همان نود رای گیری انجام داده و نتیجه را اعلام می‌کند.

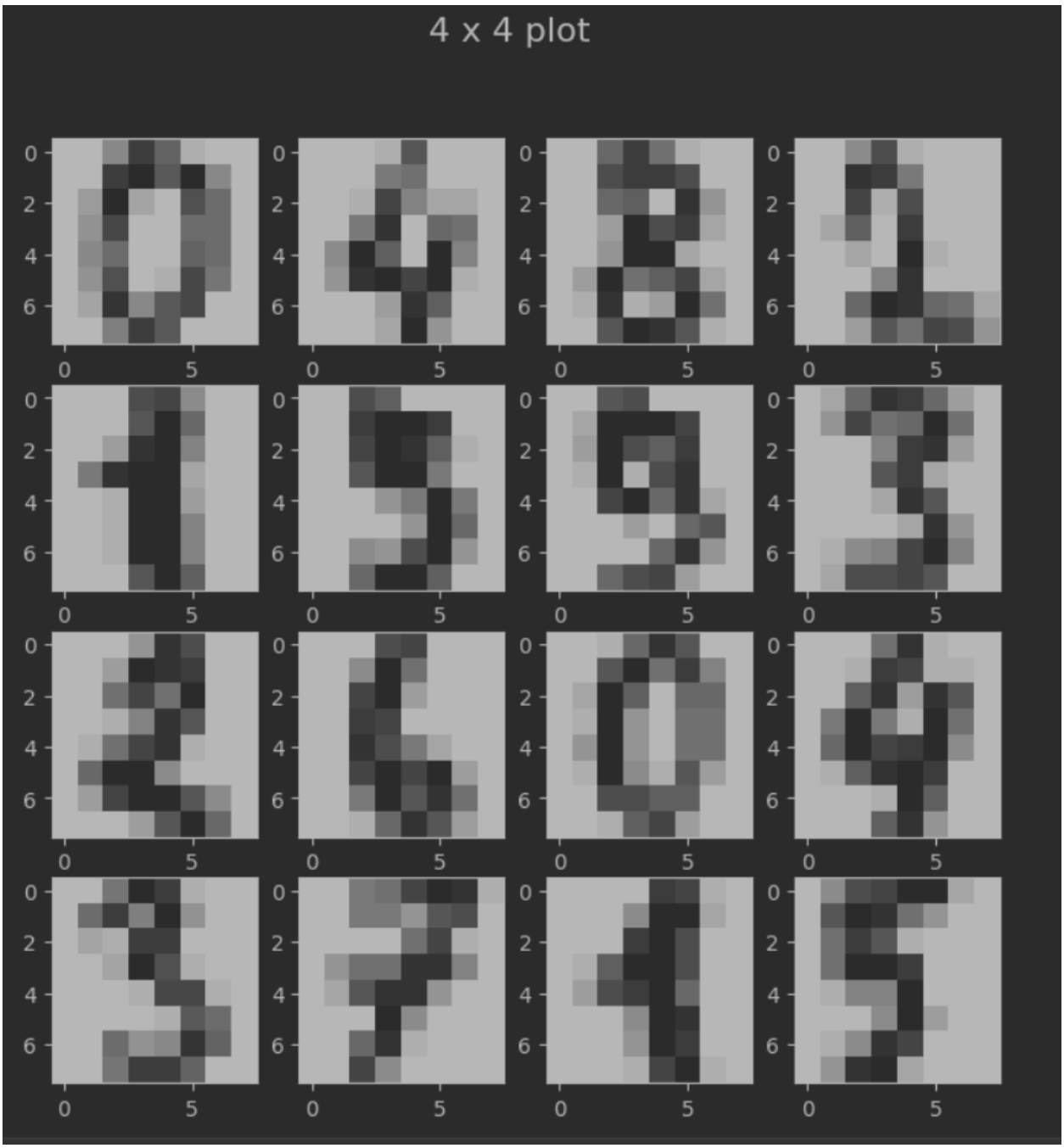


نتایج حاکی از این است که بهترین نتیجه بدست آمده در عمق ۶ بوده است که بهترین precision , accuracy برای الگوریتم ما بدست آمده است.

Q4)

a)

نتیجه اجرا بخش a



b)

در این بخش تابع KNN پیاده سازی شده است لازم به ذکر است که تابع KNN به شکل تابع طراحی شده است که تعریف فاصله مورد نظر را به عنوان ورودی می‌گیرد و می‌تولند هر معیاری که مد نظر دارید را به عنوان ورودی به آن پاس دهید که با فاصـــله اقلیدسی و کسینوسی را تست کردیم

شکل زیر نحوه فراخوانی تابع را برای تست این بخش نشان می‌دهد(عدد نوشته شده accuracy می‌باشد)

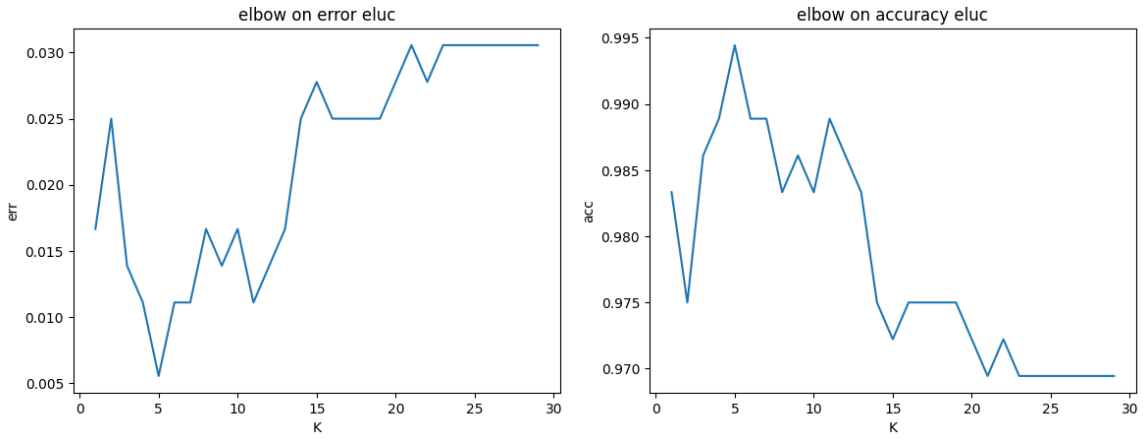
```
In 11
k = 10
y_pred = []
for i in range(len(X_test)):
    pred, _ = K_NN_estimator(k=k, X_train=X_train, y_train=y_train, distance_func=elucidation_dist,
                             input_vector=X_test[i])
    y_pred.append(pred)

accuracy_score(y_pred, y_test)

Out 11
0.9833333333333333
```

```
In 44
k = 10
y_pred = []
for i in range(len(X_test)):
    pred, _ = K_NN_estimator(k=k, X_train=X_train, y_train=y_train, distance_func=cosin_dist,
                             input_vector=X_test[i])
    y_pred.append(pred)

accuracy_score(y_pred, y_test)

Out 44
0.9888888888888889
```

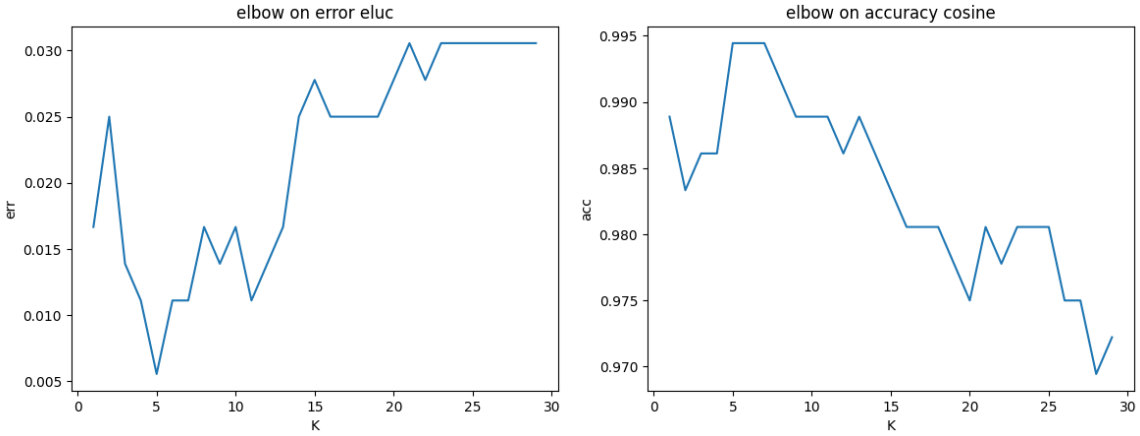
c)

در این بخش به دنبال روش elbow بودیم اما شکل نمودار به نحوی نبود که بتوانیم در نمودار ها ایجاد شده elbow را پیدا کنیم اما در هر دو مدل پیشنهادی k = 5 بهترین k مورد نظر بود. (برای هر یک شباهت کوسینوسی و اقلیدسی دو نمودار کشیده شده که یکی دقت و یکی ارزو می‌باشد که این دو نمودار به نوعی مکمل یک دیگر می‌باشند از لحاظ عددی)

فاصله اقلیدسی



فاصله کوسینوسی



d)

عملکرد هر دو الگوریتم در دیتاست موجود بسیار بسیار نزدیک به ۱ بوده است و هردو الگوریتم به خوبی عمل کرده اند اما به طور کلی cosine مقدار اندکی در همه k ها بهتر عملکرده است برای مثال در k = 5 که بهترین عملکرد هر دو مدل می‌باشد cosin حدود ۰.۰۰۶ عملکرد بهتری داشته است.

دقت cosine

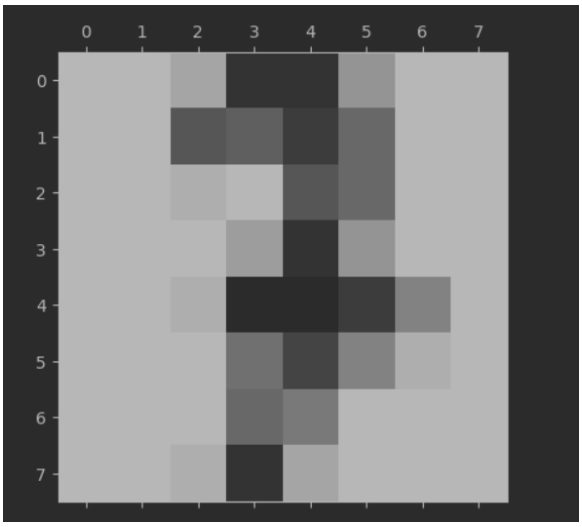
```
best k is 5
best accuracy 0.9888888888888889
```

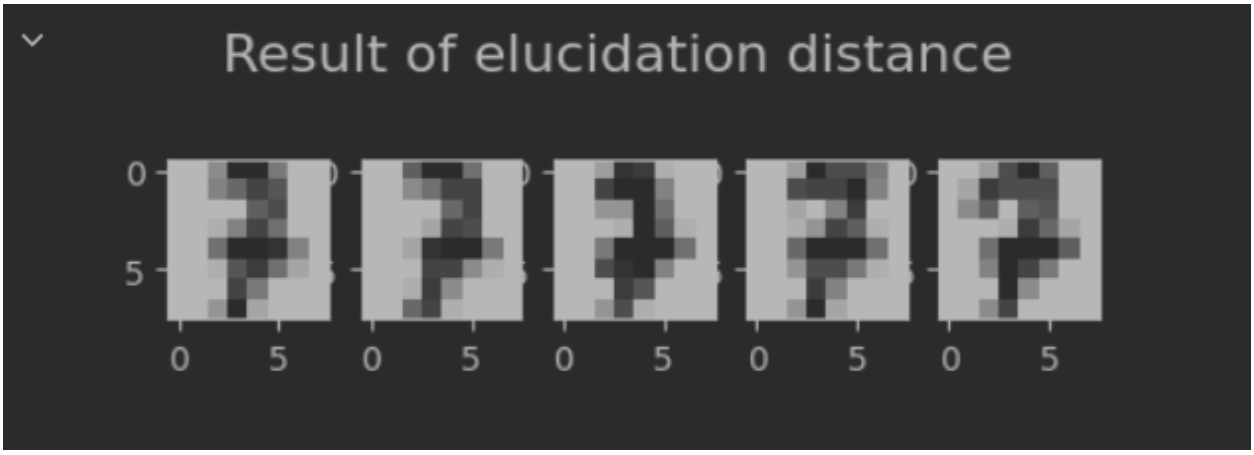
دقت مقلیدسی

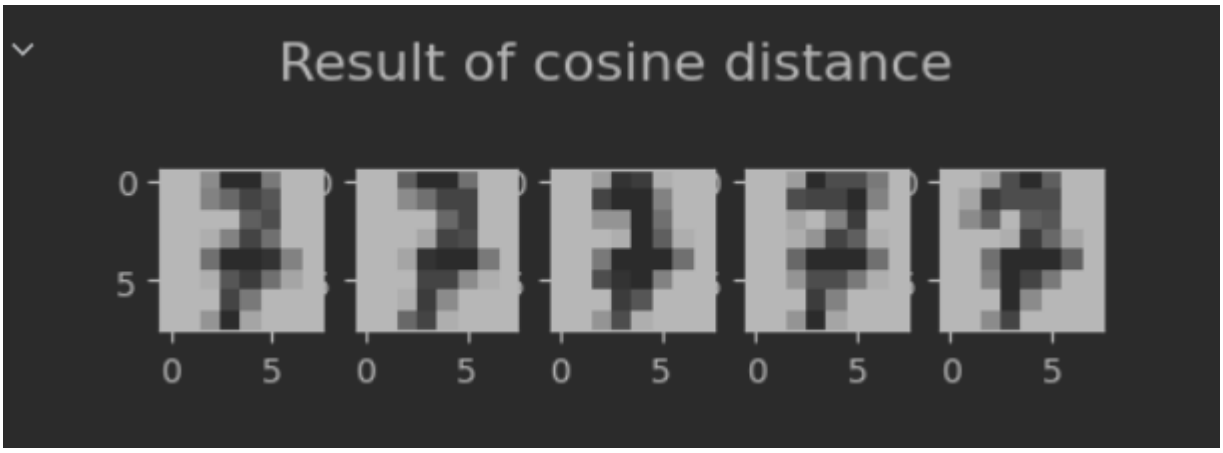
```
best k is 5
best accuracy 0.9944444444444445
```

e)

همه ۵ پیشنهاد برای عدد رندوم مورد نظر صحیح و دقیق انتخاب شده اند در هر دو معیار و الگوریتم به بهترین نحو پاسخ می‌دهد.


0 1 2 3 4 5 6 7
0
1
2
3
4
5
6
7


Result of elucidation distance
0
5
0 5 0 5 0 5 0 5 0 5


Result of cosine distance
0
5
0 5 0 5 0 5 0 5 0 5

f)

برای k = 5 این نمودار ها ترسیم گردیده اند

Elucidation distance confusion

Prediction Correct	0	1	2	3	4	5	6	7	8	9
1	0	39	0	0	0	0	0	0	0	0
2	0	0	36	0	0	0	0	0	0	0
3	0	0	0	35	0	0	0	0	0	0
4	0	0	0	0	31	0	0	0	0	0
5	0	0	0	0	0	38	0	0	0	1
6	0	0	0	0	0	0	42	0	0	0
7	0	0	0	0	0	0	0	37	0	0
8	0	0	0	0	0	0	0	0	32	0
9	0	0	0	0	0	1	0	0	0	34

10 rows × 10 columns Open in new tab

Cosine distance confusion

Prediction Correct	0	1	2	3	4	5	6	7	8	9
1	0	39	0	0	0	0	0	0	0	0
2	0	0	36	0	0	0	0	0	0	0
3	0	0	0	35	0	0	0	0	0	0
4	0	0	0	0	31	0	0	0	0	0
5	0	0	0	0	0	38	0	0	0	1
6	0	0	0	0	0	0	42	0	0	0
7	0	0	0	0	0	0	0	37	0	0
8	0	0	0	0	0	0	0	0	32	0
9	0	0	0	0	0	1	0	0	0	34

10 rows × 10 columns Open in new tab

g)

در این بخش کد مربوط به هر یک از ویژگی ها نوشته شده

برای مقدار ۸

```
give_all_info(eluc_confusion,8)

TP for 8 => 32
FP for 8 => 0
FN for 8 => 0
F1_score for 8 => 1.0
```

برای مقدار ۳

```
give_all_info(eluc_confusion,3)

TP for 3 => 35
FP for 3 => 0
FN for 3 => 0
F1_score for 3 => 1.0
```

h)

برای مقدار ۶

```
give_all_info(cosine_confusion,6)

TP for 6 => 42
FP for 6 => 0
FN for 6 => 0
F1_score for 6 => 1.0
```

برای مقدار ۴

```
give_all_info(cosine_confusion,4)
```

```
TP for 4 => 31
FP for 4 => 0
FN for 4 => 0
F1_score for 4 => 1.0
```

از آنجایی که الگوریتم در هیچ یک از اعداد بالا خطایی نکرده است برای عدد ۹ نیز اجرا کردیم

```
give_all_info(cosine_confusion,9)
```

```
TP for 9 => 34
FP for 9 => 1
FN for 9 => 1
F1_score for 9 => 0.9714285714285714
```