به نام خدا

دانشگاه صنعتی امیرکبیر (پلی تکنیک تهران)

دانشکده مهندسی کامپیوتر

یادگیری ماشین

تکلیف دوم

استاد درس: دکتر ناظرفرد

امیرحسین کاشانی

۴۰۰۱۳۱۰۷۱

amkkashani@gmail.com

نیم سال اول ۱۴۰۱-۱۴۰۲

فهرست

Q1)

a)

a) انواع فاصله (۳ مورد مطرح شده در سوال)

- Euclidean (فاصله اقلیدسی)

این فاصله برابر طول یا اندازه خطی است که بین دو نقطه قرار می‌گیرد که برای دو نقطه‌ی p,q عبارت است از (n بعدی)

$$d(p,q) = \sqrt{(p_1-q_1)^2+(p_2-q_2)^2+\dots+(p_n-q_n)^2}$$

$$= \sqrt{\sum_{i=1}^{n}(p_i-q_i)^2}$$

- Cosine (کوسینوسی)

نکته‌ای که در این مورد باید به آن توجه شود این است که در این جا Cosine similarity و Cosine distance بردی این معیار تعریف می‌شوند که یکی شباهت و دیگری تفاوت را به ما نشان می‌دهد Cosine Similarity = 1 − Cosine distance

$$\text{Cosine Similarity} = \frac{A.B}{|A|.|B|}$$

همانطور که از فرمول معلوم است این روش به زاویه‌ی بین دو بردار نگاه می‌کند و مقادیر اندازه در صورت scale شدن تأثیری نخواهد داشت و یکی از معایبی

که به این روش گرفته می‌شود این است که برداری که از درجه با آن اختلاف دارد دورتر از برداری است که ۱۰۰ برابر برداری است که در همان زاویه است البته این موضوع به خودی خود مشکل به حساب نمی‌آید بلکه در موضوع مسأله باید طراحی گردد.

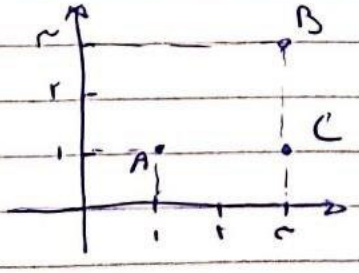
- City Block

در این معیار، فاصله عبارت است از قدر مطلق اختلاف در هر یک از ابعاد

$$D(x,y) = \sum_{i=1}^{k} |x_i - y_i|$$

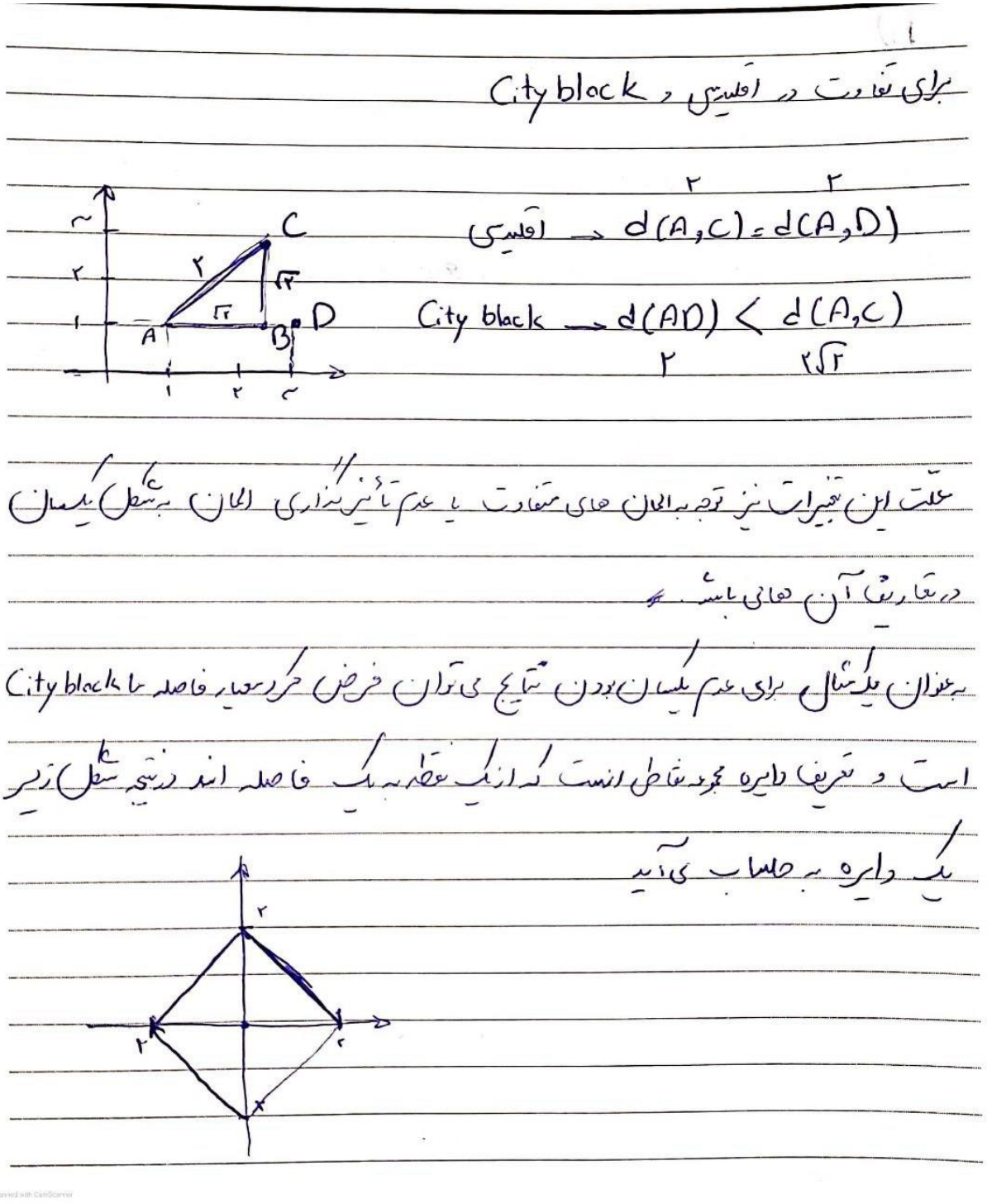
این معیار در مقایسه با فاصله‌ی اقلیدسی مقادیر بزرگتری را به عنوان خروجی تولید می‌کند، از نظر محاسباتی ساده‌تر است و در ابعاد زیاد

* در رابطه با مقادیر خروجی نیز باید گفت هر ۳ در حالت‌های متفاوت ممکن است خروجی متفاوت دهند برای مثال Cosine با دو معیار دیگر می‌آن مثال زیر را بیان کرد.



Cosine → d(AB) < d(AC)

Eucl, Manhattan → d(AC) < d(AB)



b)

با توجه به ماهیت مساله و داده های ورودی ما از آنجا که روشنایی تصویر در دسته بندی و تشخیص چهره نباید تاثیری در فاصله داشته باشد(cosine به scale توجه نمی‌کند و درنتیجه شدت نور را در نظر نمی‌گیرد و در مقابل فاصله اقلیدسی و منهتن هر دو واکنش شدیدی به این موضوع دارند) فاصله کوسینوسی عملکرد بهتری در تصاویر خواهد داشت. به عنوان یک دلیل نه چندان

محکم دیگر نیز می‌توان این را بیان کرد که cosine عملکرد بهتری در زمانی که ویژگی ها نرمال نشـده باشـند دارد.(این امر باعث می‌شود زمانی که تصاویر مثلا از یک فیلتر سبز رنگ عبورکنند کارآیی فاصله کسینوسی بهتر از دو معیار دیگر باشد)

داده های بدست آمده در سوال ۴ پیاده سازی نیز گواه این موضوع بوده اند.

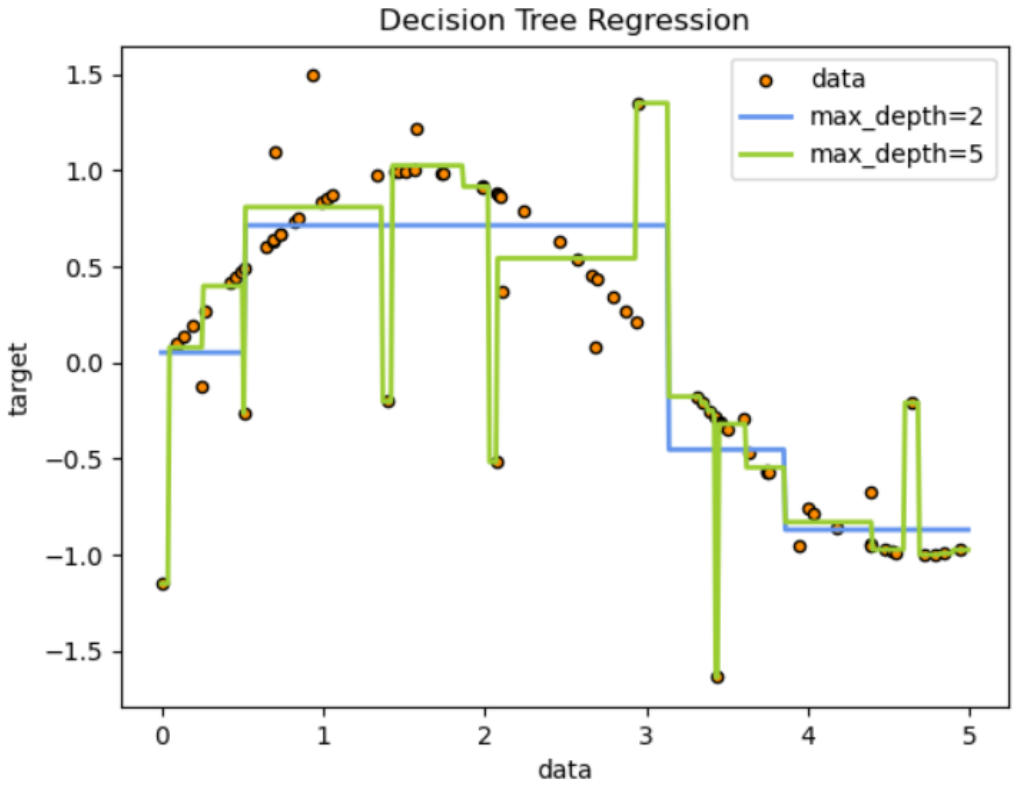
c)

KNN

در رابطه با اسـتفاده از KNN در رگرسـیون می‌توان گفت در این مسـائل به دنبال تخمین مقدار y هسـتیم (y یک کمیت پیوسـته می‌باشـد) الگوریتم KNN را اجرا می‌کنیم و K همسـایه نزدیک در feature های موجود را پیدا میکنیم سـپس بین آن ها میانگین می‌گیریم. در حالات خاص می‌توانیم از میانگین وزن دار بر اساس فاصله نیز استفاده کنیم اما در الگوریتم پایه صرفا میانگین گرفته می‌شود.

Decision tree

این روش بیشـتر برای تحلیل منحنی هایی به کار می‌روند که دارای نویز می‌باشـند و به این صـورت عمل می‌کنند که ابتدا درخت تصمیم بر اساس ویژگی های موجود فضای حالت را مانند شکل موجود به قسمت های متفاوت تقسیم می‌کند و سپس در قسمت برگ برای آن نقاط الگوریتم رگرسـیون را اجرا می‌کند. از معادله خط بدسـت آمده برای پاسـخ دادند به داده های تسـت که در آن ناحیه می‌باشند استفاده خواهد شد.



d)

در فرآیند ساخت درخت هرچه عمق درخت بیشتر باشد تصمیم گیری صورت گرفته به سمت بیش برازش یا overfit میل می‌کند به بیان دیگر عمیق شـــدن یک درخت به معنی توجه بیش از حد به جزئیات داده های آموزش می‌باشـــد که این امر عملکرد کلی درخت را کاهش می‌دهد.

پیش هرس(pre prune)

این نوع هرس در زمان ساخت درخت صورت می‌گیرد که مثال های به اختصار ذکر می‌گردد

- محدود کردن عمق، به عنوان مثال اگر به عمق ۱۰ رسیدیم در هر وضعیتی که بود تقسیم نود را خاتمه دهیم و براساس اکثریت رای گیری کنیم
- بررسی میزان خلوص و فرض کردن یک threshold که در صورتی که میزان خلوص نود ما از آن مقدار بیشتر بود دیگر به عمل تقسیم ادامه ندهیم برای مثال اگر ۰.۹ یا بیشتر از دسته ما یک لیبل داشتند دیگر نیازی به ادامه تقسیم نود ها نباشد

پس هرس (post prune)

در این سبک از هرس درخت ما از قبل ساخته شده است و ما می‌خواهیم در زمان prediction هرس را انجام دهیم. در این نوع هرس در اصل تصمیم می‌گیریم چه میزان در درخت تصمیم به عمق درخت نفوذ کنیم در ادامه چند راه حل بیان شده را می‌بینیم

- مانند پیش هرس عمق را محدود کنیم یعنی با وجود اینکه درخت اعماق بیشتری برای جست و جو را پشتیبانی می‌کند اما ما مثلا حداکثرتا عمق ۵ پیش برویم و در آن نود رای گیری انجام دهیم
- روش دوم روشی جامع تری می‌باشد و از معیار cost complexity pruning استفاده می‌شود که از رابطه زیر بدست می‌آید

$$\frac{\mathrm{err}(\mathrm{prune}(T,t),S)-\mathrm{err}(T,S)}{|\mathrm{leaves}(T)|-|\mathrm{leaves}(\mathrm{prune}(T,t))|}$$

این رابطه به ما نشـــان می‌دهد که آیا با تقســـیم این نود و ادامه دادن این درخت چه میزان بهبود در خطای موجود درخت داده می‌شـــود. با قرار دادن یک threshold بر روی این معیار در هر عمقی از درخت می‌توانیم زمانی که ایجاد تغییرات دیگر چند فایده ای برای دقت مساله نداشت تقسیم دسته ها را خاتمه دهیم و همان نود را به عنوان برگ در نظر بگیریم.

e)

e) $Entropy(p) = -(P\log_2 P + (1-P)\log_2(1-P))$

$Gain = E_{Parents} - \sum \frac{n}{N} E_{child}$

$E_{Parent} = E(0.(V0) = .1\Lambda1$

F_1: $+ \rightarrow E(1) = 0$, $- \rightarrow E(1) = 0$ $\rightarrow Gain(F1) = .1\Lambda1$

F_2: $+ \rightarrow E(1) = 0$, $- \rightarrow E(1) = 0$ $\rightarrow Gain(F2) = .1\Lambda1 - 0 = .1\Lambda1$

$F_3 = \begin{cases} + \rightarrow E(1) = 0 \\ - \rightarrow E(\frac{2}{4}) = 0.191 \end{cases} \rightarrow Gain(F3) = .181 - .191 \times \frac{3}{4} = 0.127$

$F_5 = \begin{cases} + \rightarrow E(\frac{2}{5}) = 0.191 \\ - \rightarrow E(1) = 0 \end{cases} \rightarrow Gain(F5) = .127$

$F_6 = \begin{cases} + \rightarrow E(.125) = .181 \\ - \\ - \rightarrow E(0) = 1 \not\Phi \end{cases} \quad Gain F_6 = 0$

$F_4 = \begin{cases} + \rightarrow E(.010) = 1 \\ - \rightarrow E(1) = 0 \end{cases} \quad Gain F_4 = .181 - .010 = .171$

دراین قسمت F_1 و F_2 بهترین ویژگی ها برای قرار گرفتن در ریشه ی درخت بودند و به دلیل بیشترین Gain به دست آمده

f)

f)

$E_{Parent} = 0.19V$

$F1 = \begin{cases} + \rightarrow E(.10) , E(.170) = 0.181 \\ - \rightarrow E(1) = 0 \end{cases}$ Gain

$= 0.19V - \frac{3}{4} \times 0.181 = 0.136$

$F_2 = \begin{cases} + \rightarrow E(1) = 0 \\ - \rightarrow E(1) = 0 \end{cases} \rightarrow Gain = 0.19V$

باتوجه به اینکه F_2 بالاترین خلوص عمل را دارد دقیقه ی دسته ها دارای ناخالصی

هستند جواب نهایی F_2 می باشد

g)

Train and test time complexities

Train time complexity:

$$O(n * logn * d)$$

$$n: data\ points$$
$$d: dimensions$$

Figure 31

What happens in the training stage is that for each of the features (dimensions) in the dataset we'll sort the data which takes O(n log n) time following which we traverse the data points to find the right threshold which takes O(n) time. For d dimensions, total time complexity would be:

$$O(nlogn * d) + O(n * d)$$

which asymptotically is

$$O(nlogn * d)$$

در هر سطر درخت ما باید همه n داده بررسی شوند حتی در حالاتی که در نود های متفاوت باشند. برای محاسبه انتروپی وgain نیاز است که همه ویژگی ها بررسی گردد که تا اینجا سبب می‌گردد که در هر سطر پیچیدگی زمان O(n * d) داشته باشیم. عمق درخت نیز log n می‌باشد (البته این فرض خوش بینانه است) و در نهایت o(n * log(n) * d) حد بالای ما خواهد شد. در صورت بالانس نبودن درخت ممکن است که به $O(n^2 * d)$ برخورد کنیم که بدلیل پایین بودن احتمال این موضوع و صرف نظر شدن از این حالت در منابع از آن عبور می‌کنیم.

در نتیجه طبق فرمول بالا ، با فرض اینکه عمق درخت ثابت است نسبت زمان حالت جدید به حالت قبلی برابر

$$10\, n \log(10 * n) * d = 13.32\, n \log(n) * d$$

همانطور که می‌دانیم این صـرفا یک تقریب اسـت و نمی‌توانیم صـرفا از روی O یک مقدار دقیق را حدس بزنیم و در صـورتی که بخواهیم تقریب دقیق تر داشـته باشـیم باید امید ریاضـی ایجاد درخت ها را محاسـبه کنیم که بدلیل پیچیدگی و خارج شـدن از موضوع به همین تقریب بسنده شده است. مدت زمان شده با در نظر گرفتن log در مبنای ۲ برابر حدود ۱۳ ساعت و نیم می‌باش‌د.

h)

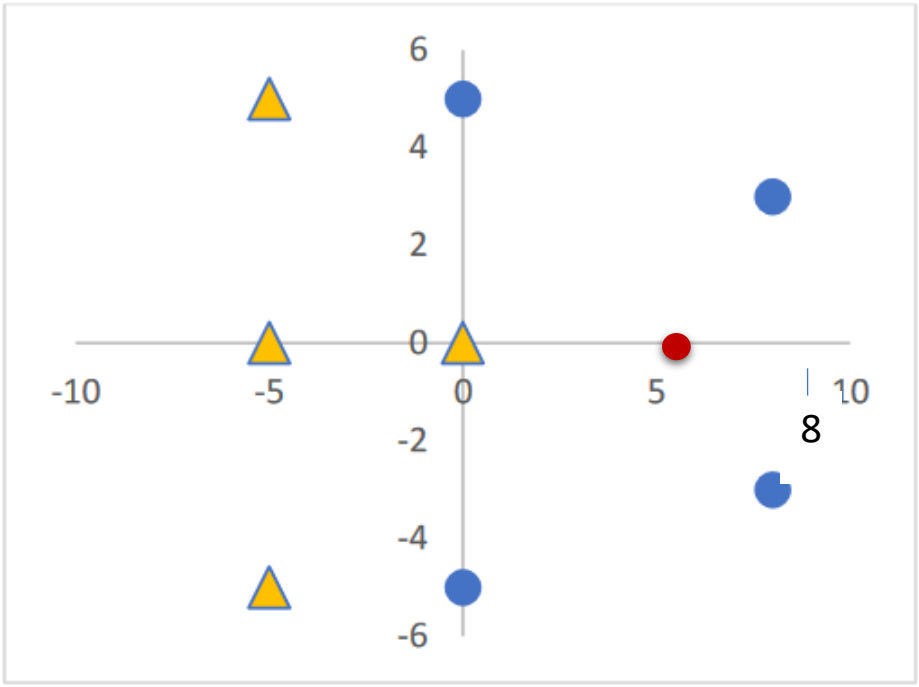
در یک درخت باینری با عمق m ، 2^m برگ خواهیم داشت (نود اولیه عمق صفر در نظر گرفته شده است) و درخت متوازن درختی است که بیشـترین نود ممکن را برای ما ایجاد می‌کند، در نتیجه تعداد برگ های ما در بیشـترین حالت برابر 2^h که h عمق درخت می‌باشـد، اسـت. اما در اینجا دو حالت پیش می‌آید حالتی که تعداد داده های کمتر از 2^m باشـد که در این حالت تعداد داده ها محدود کننده اسـت و عمق از رابطه $h = \log n$ بدسـت می‌آید و حالت دیگر این که داده ها به میزان قابل توجه بیشـتر از 2^m باشند که در اینجا تعداد ویژگی ها محدود کننده هستند و $h= m$ می‌شود.

I)

برای $[5,0]^T$

با معیار d1 نقطه مورد نظر با قرمز مشخص شده است می‌بینیم که از نظر فاصل در y با اینکه با نود زرد رنگ صفر است اما فاصله x آن ها بیشـتر اسـت و نسـبت به نود آبین که حدود ۳و۳در محور x , y از آن فاصـله دارند دور تر در نظر گرفته می‌شـود **درنتیجه معیار d1 این را آبی تشخیص می‌دهد.**

با معیار d2 می‌توانیم فرض کنیم که نقاط آبی در مختصـات ۳و۳ نسـبی به نود قرمز (نود ورودی) قرار دارند که درنتیجه فاصـله آن ها برابر 6 می‌شود و نود زرد رنگ در فاصله ۵ قرار دارند و **نود ورودی زرد پیش بینی خواهد شد.**

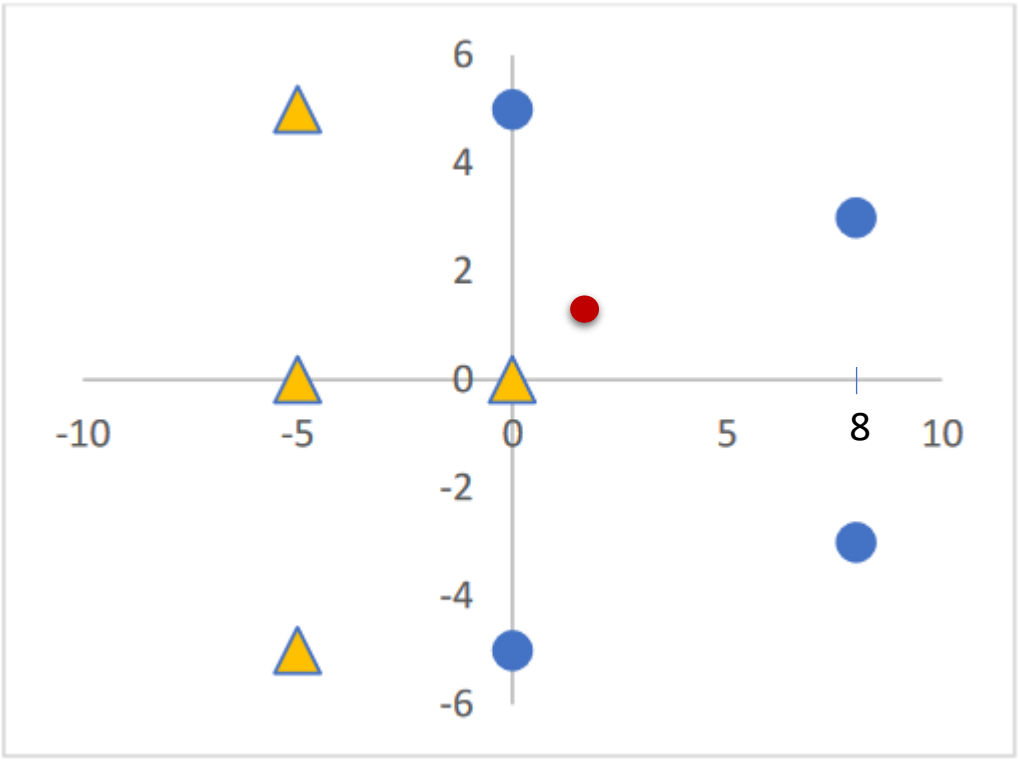


K_NN با k = 4

برای $[1,1]^T$

با معیار d1 ۳ نود زرد که در بالا سمت چپ قرار دارند و نود آبی که روی محور y در نقطه ۵ قرار گرفته انتخاب می‌شود و درنتیجه پیش بینی زرد خواهد بود

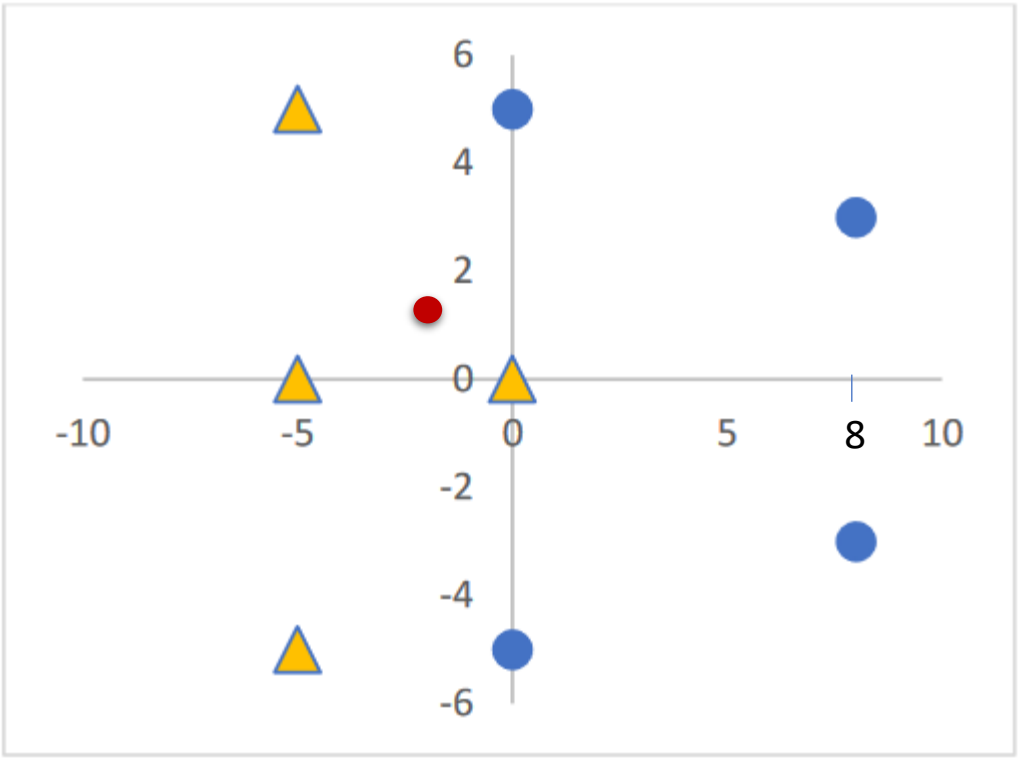
با معیار d2 نتیجه 2 نود زرد (که روی محور x هستند) و دونود آبی که یکی روی محور y و دیگری در ناحیه یک مختصات می‌باشد است در این حالت رای ها بصورت مساوی می‌باشد و می‌توانی یکی از دو رنگ را بصورت رندوم انتخاب کرد



برای نقطه $[-2,1]^T$ (تصویر درپایی آمده است)

با معیار d1 مشابه d1 بخش قبل ۳ زرد و یک آبی و نتیجه زرد خواهد بود

با معیار d2 در ۳ نود زرد (نود های روی محور x و نودی که در ناحیه دو قرار دارد) انتخاب می‌شود و همچنین یک نود آبی که روی محور y است و نتیجه زرد خواهد بود

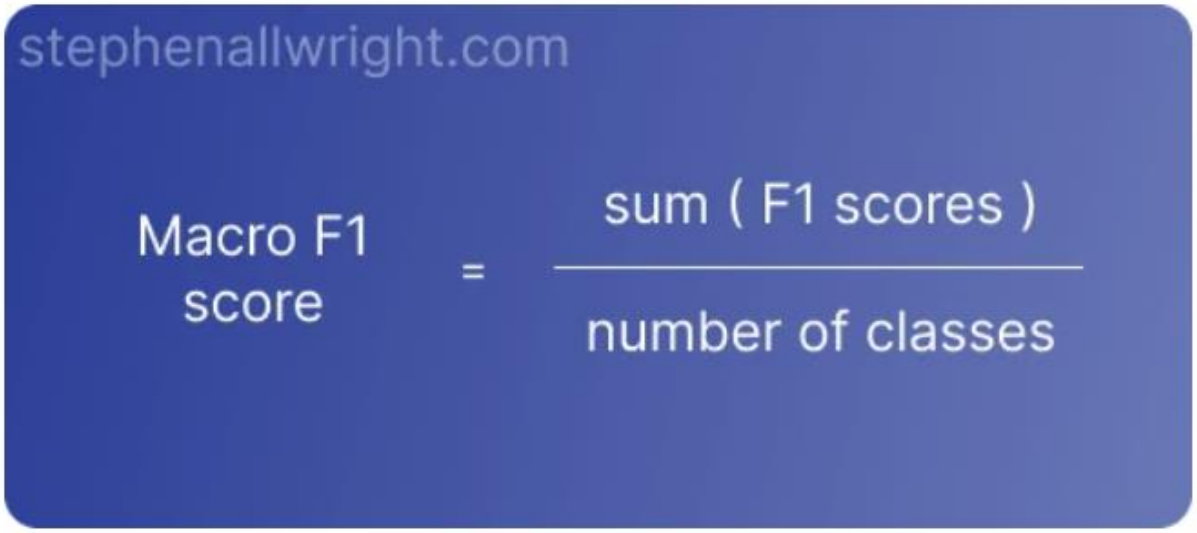


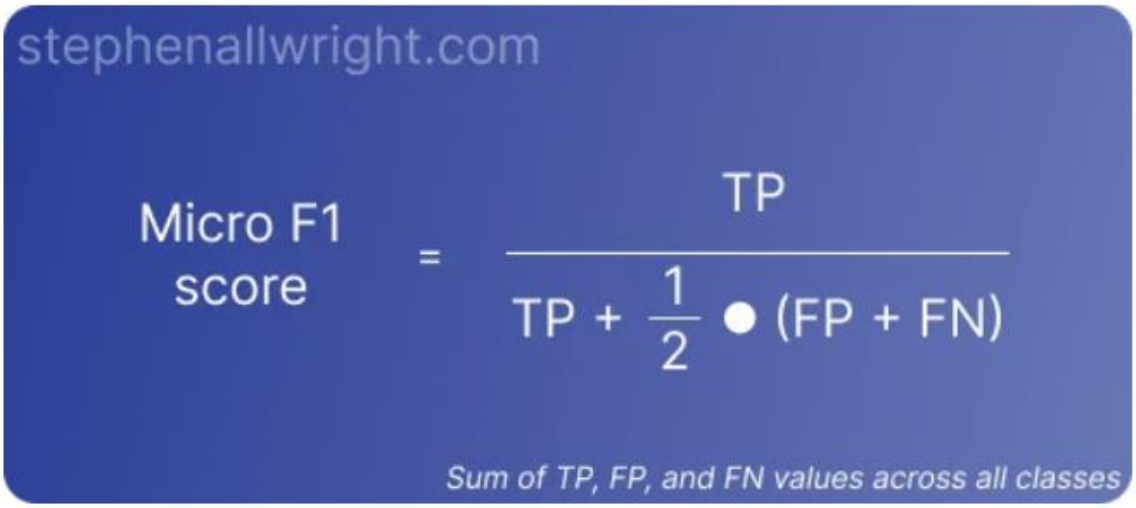
j)

تفاوت micro و macro

در داده هایی که imbalance هستند(با معیارF1) عملکرد micro ضعیف تر از macro می‌باشد به این دلیل که micro F1 اهمیت یکسانی به هریک از داده ها می‌دهد ولی macro به هر یک از دسته ها اهمیت یکسانی می‌دهد.

در micro F1 کلاس بزرگتر اهمیت بیشتری پیدا می‌کند برای مثال اگر ۹۰ درصد داده ها مربوط به این کلاس باشند اکثر توجه به سمت این کلاس می‌رود و این در صورتی است که عموما در این نوع مسائل هدف جدا کردن آن ۱۰ درصد خاص می‌باشد. اما در دید macro F1 همه کلاس ها یک میزان اهمیت دارند مستقل از تعداد اعضای آن ها و اجازه در صورتی که یک کلاس با جمعیت کم عملکرد ضعیفی داشته باشد بیشتر خود را نشان خواهد داد نسبت به micro F1.

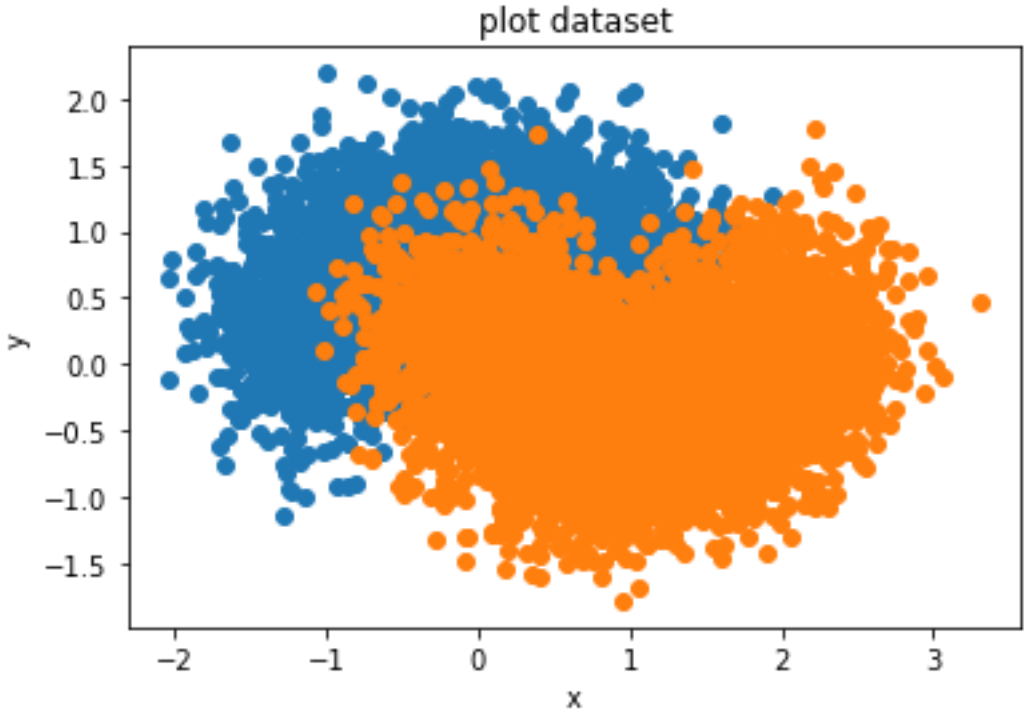

stephenallwright.com
Macro F1 score
=
sum (F1 scores)
number of classes


stephenallwright.com
Micro F1 score
=
TP
TP + 1/2 ● (FP + FN)
Sum of TP, FP, and FN values across all classes

Q2)

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 شـــده در بخش d بهتر اســـت. نتیجه میزان دقت درخت برابر ۸۵/۸ و تک درخت آموزش داده شـــده 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
```

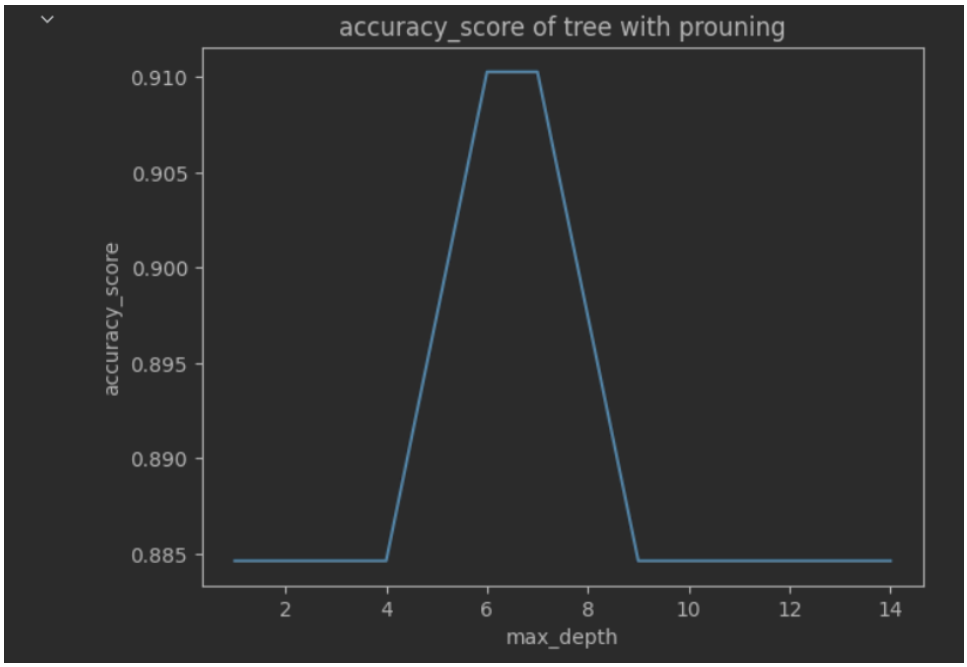
تکرار بخش b

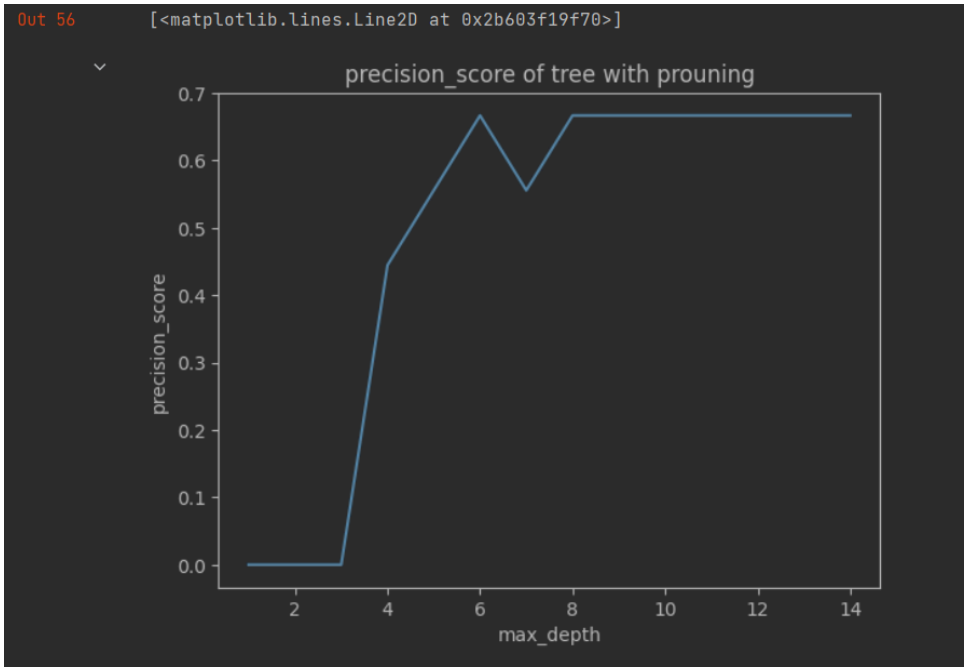
```
train_size : 0.2 => avg accuracy : 0.8605990783410139, avg precision : 0.44534152548991884
train_size : 0.45 => avg accuracy : 0.8722689075630253, avg precision : 0.6210266715381805
train_size : 0.65 => avg accuracy : 0.8951507208387942, avg precision : 0.5605302727151467
train_size : 0.85 => avg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



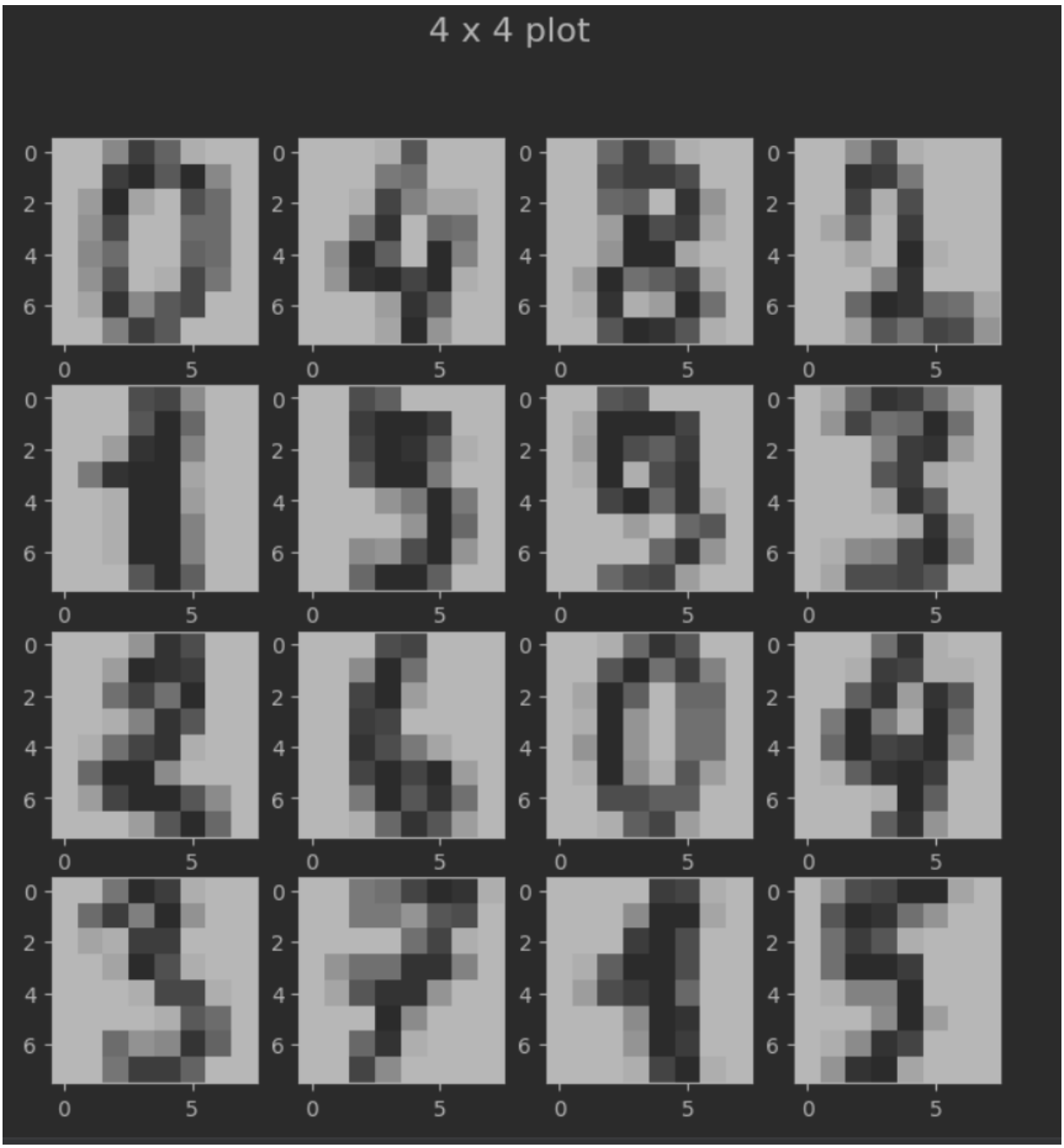


نتایج حاکی از این است که بهترین نتیجه بدست آمده در عمق ۶ بوده است که بهترین precision , accuracy برای الگوریتم ما بدست آمده است.

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)

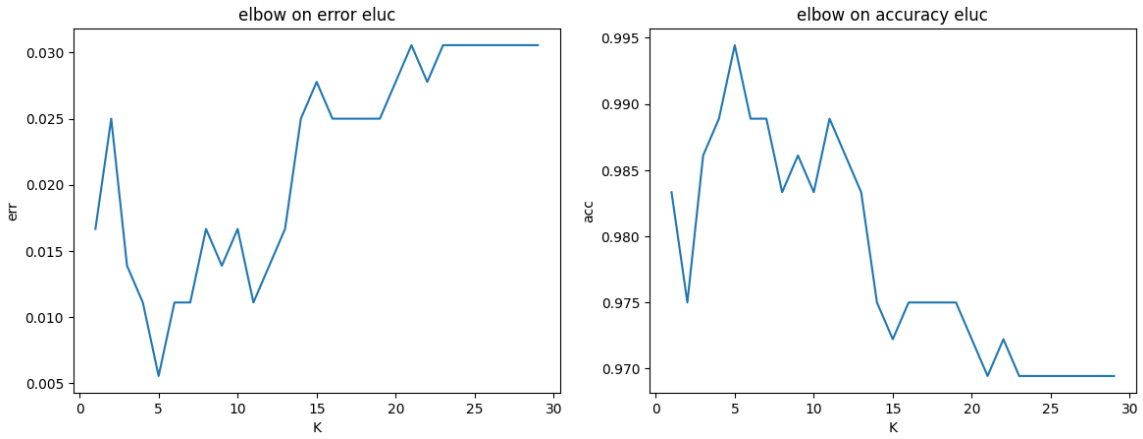
```
In 11
k = 10
y_pred = []
for i in range(len(X_test)):
    pred, _ = K_NN_estimator(k=k, X_train=X_train, y_train=y_train, distance_func=elucidation_dist,
                             input_vector=X_test[i])
    y_pred.append(pred)

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
```

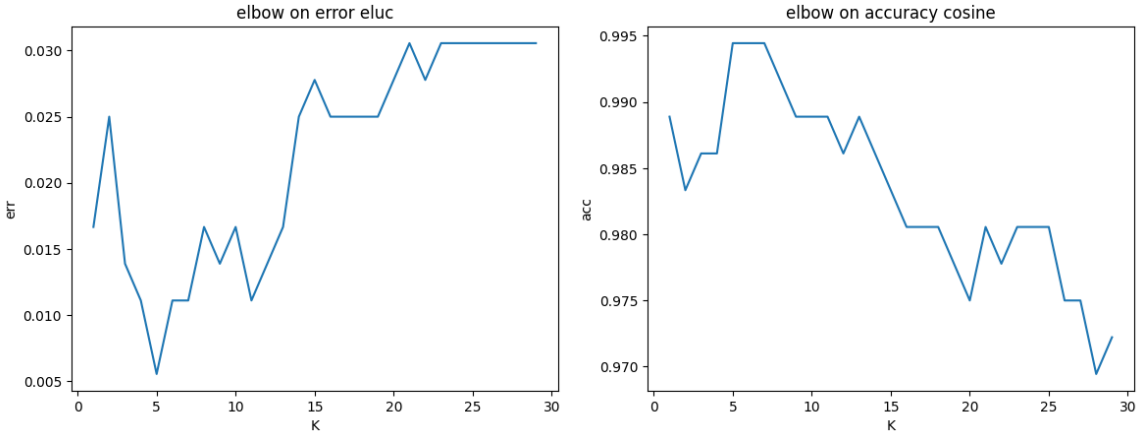
c)

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 اررو 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

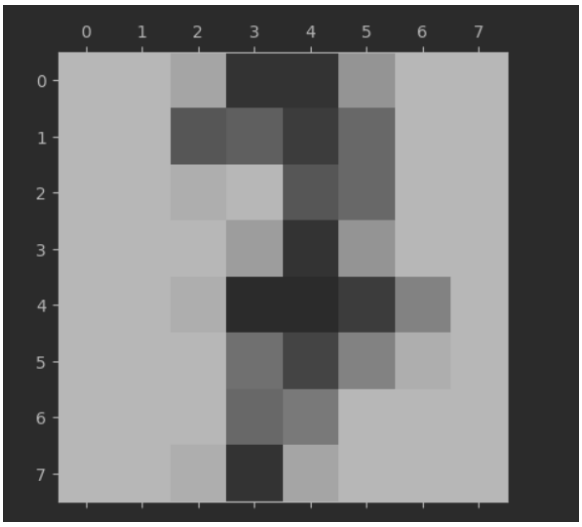
```
best k is 5
best accuracy 0.9888888888888889
```

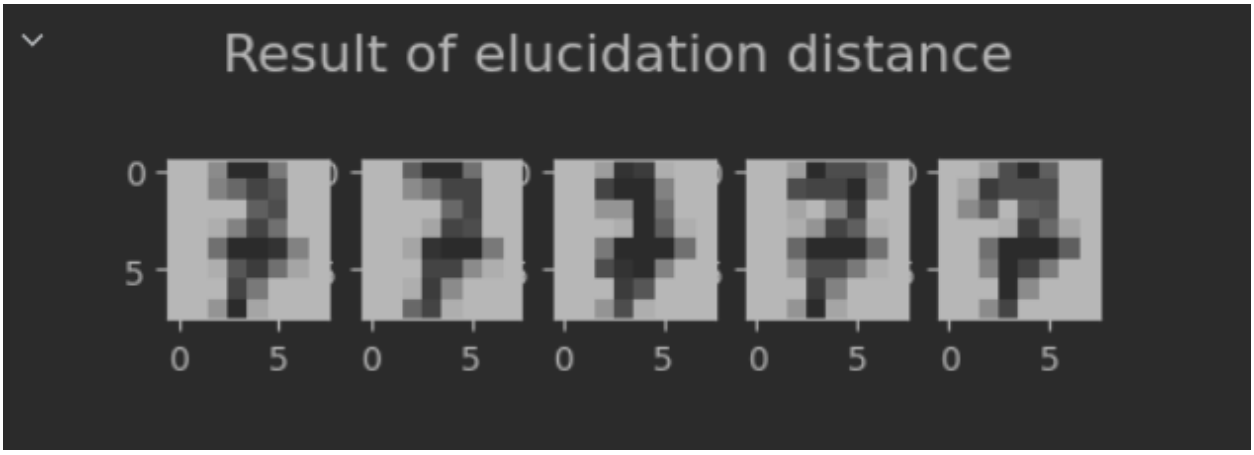
دقت مقلیدسی

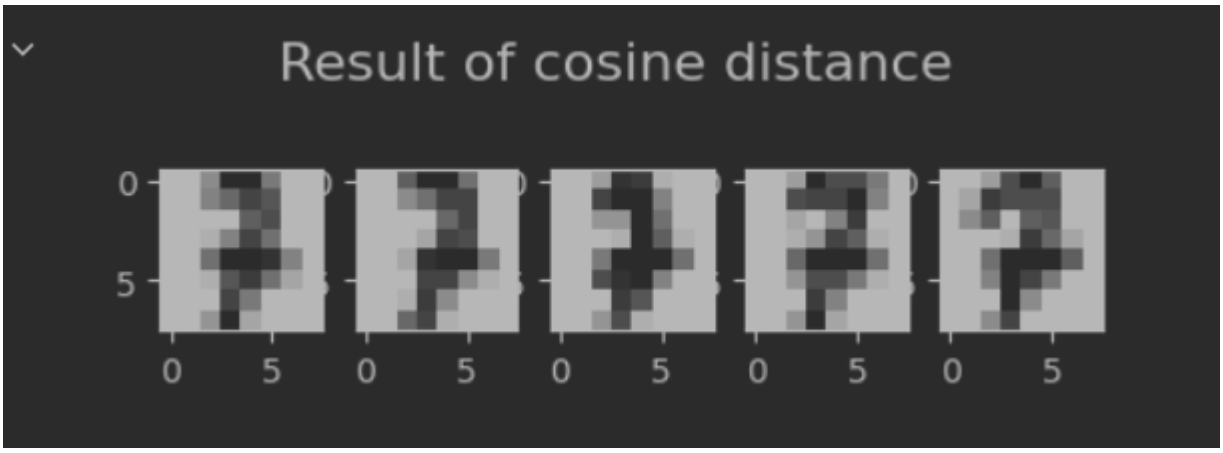
```
best k is 5
best accuracy 0.9944444444444445
```

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
```