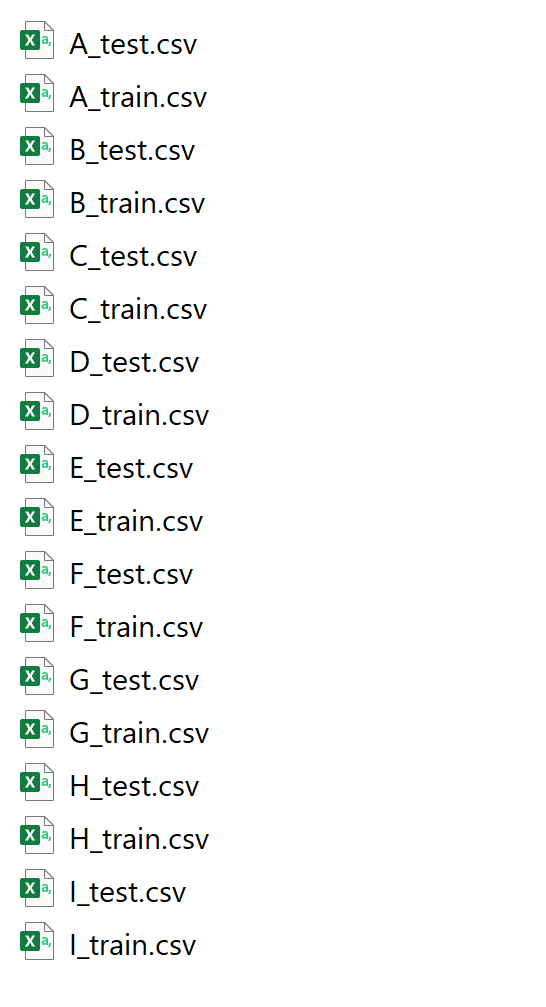
## امیررضا صدیقین

## 993614024

## سوال 1 :

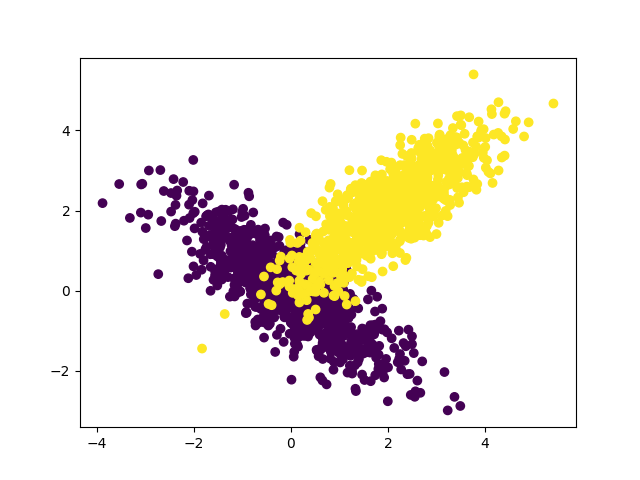
کد های مربوط به این بخش در فولدر problem1 در codeها است.

با استفاده از کد مربوط به فایل text\_to\_csv.py داده‌های لازم را منظم در CSVها ریختم که در فولدر مربوط به data در problem1 موجود است.

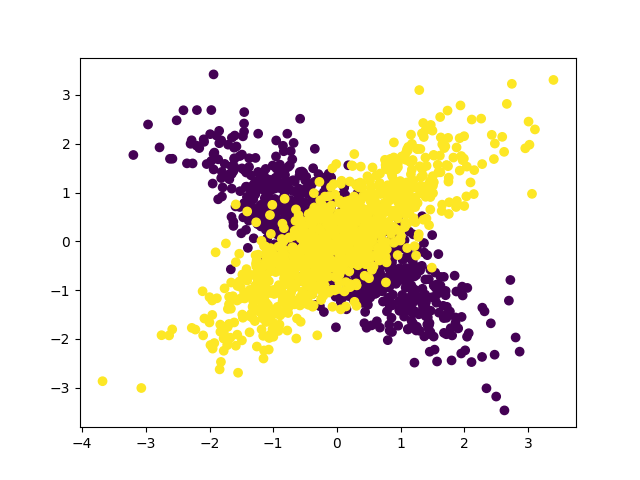


با استفاده از کد مربوط در فایل plot.py داده ها را نمایش داده ام که در result/plots عکس‌ها موجود است و به شرح زیر است.

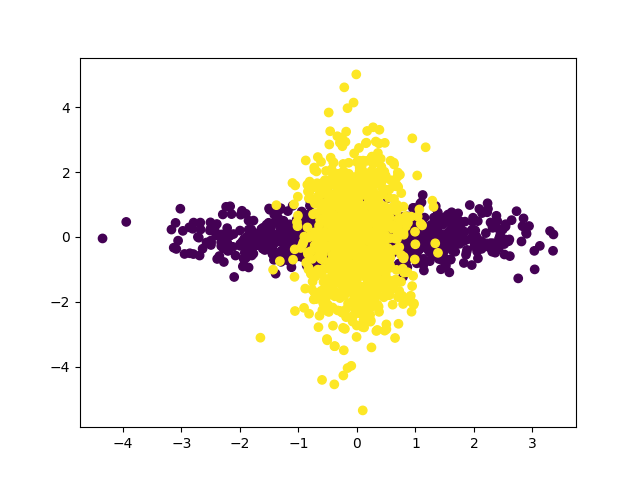
* A



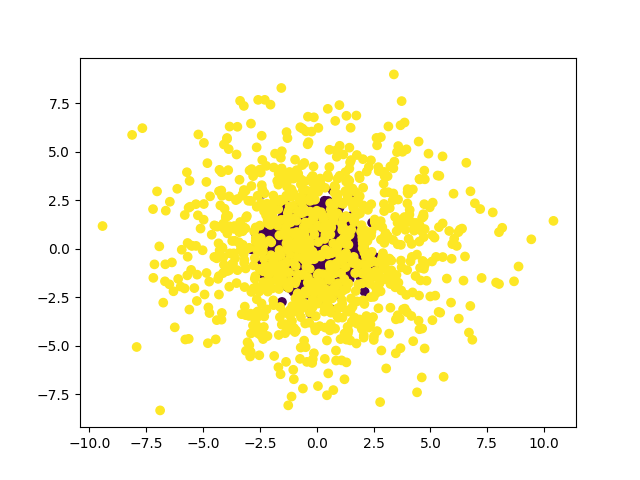
* B



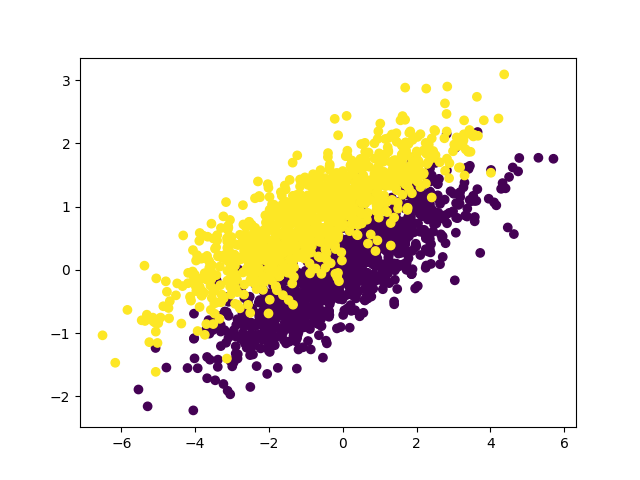
* C



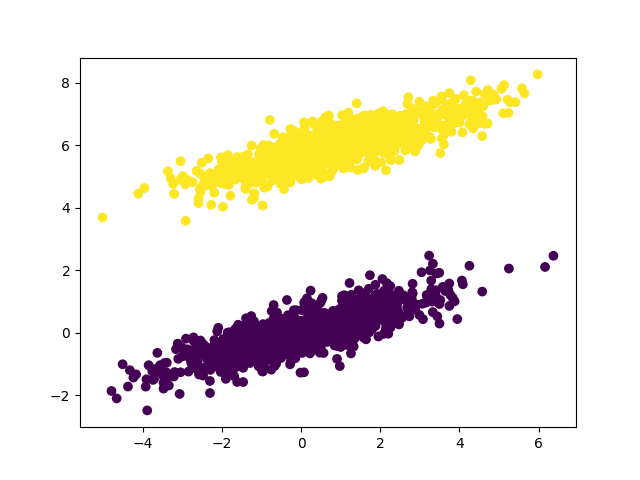
* D



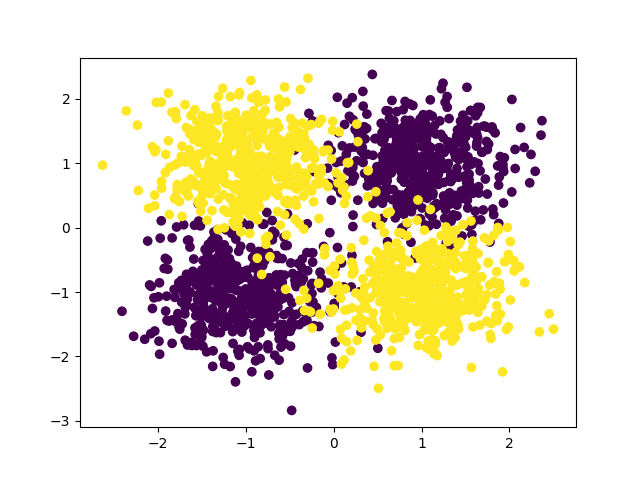
* E



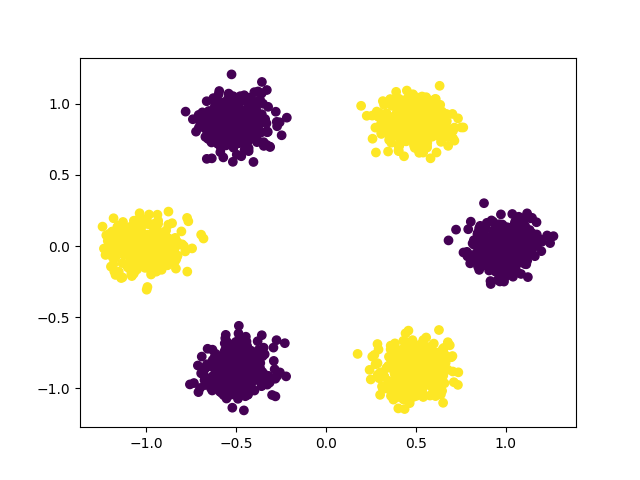
* F



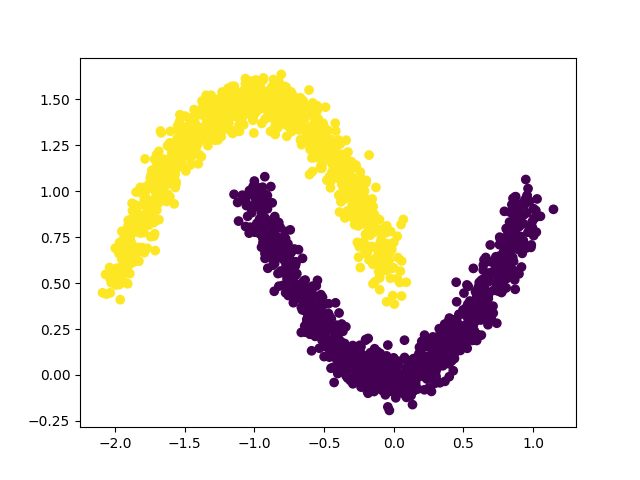
* G



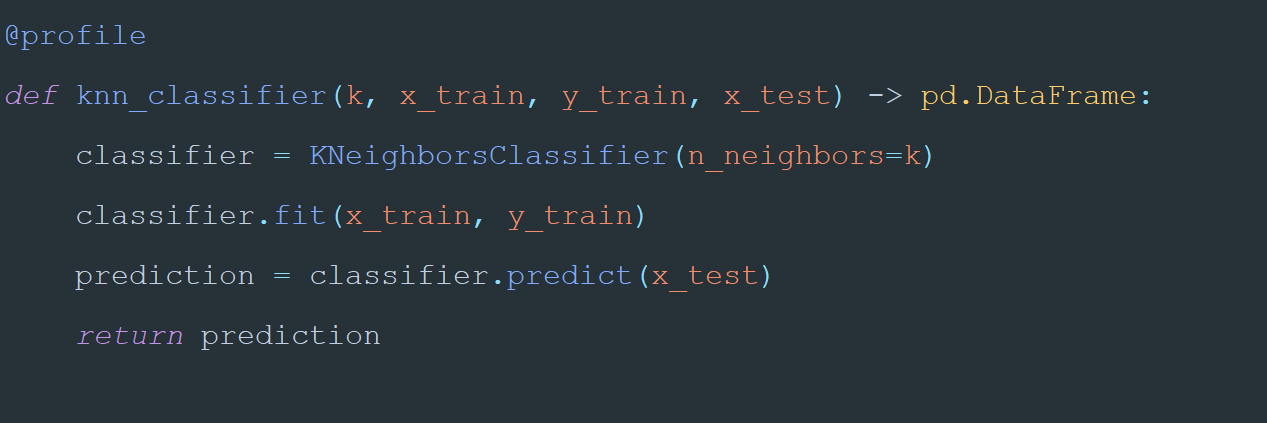
* H



* I



در فایل مربوط به classify\_knn.py تابعی به اسم knn\_classifier است که وظیفه‌ی classify کردن داده‌های train با هر ابعاد و تعداد دسته را بر عهده دارد و نیز داده‌های تست را classify میکند(predict میکند) (کد مربوط به بخش اصلی سوال)

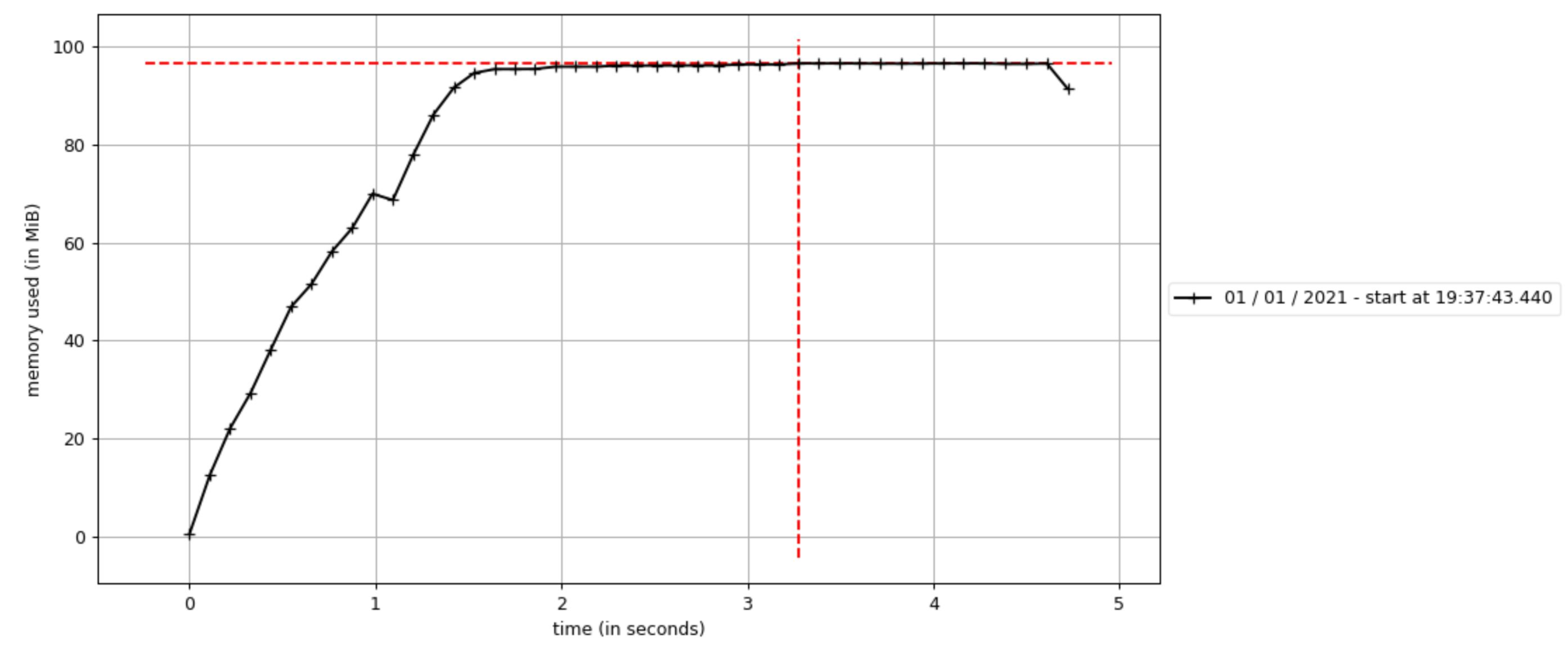


در فایل classify\_knn.py با استفاده از تابع فوق و تابع classify\_toy\_data ، این الگوریتم را بر روی هر یک از مسائل A تا I انجام دادم که دسته‌بندی داده‌های تست در فایل‌های csv مربوط به هر مسئله در فولدر /result/predictions آمده است که نشان میدهد لیبل درست و پیش بینی الگوریتم هر کدام چه بوده است. همچنین دقت و ماتریس‌های تداخل در فایل problem1\_1.json واقع در result ذخیره شده است که به شکل زیر است: (پاسخ بخش الف)

{  
 'A': {  
 'confusion\_matrix': [[980, 20],  
 [73, 927]],  
 'accuracy': 0.9535  
 },  
 'B': {  
 'confusion\_matrix': [[839, 161],  
 [253, 747]],  
 'accuracy': 0.793  
 },  
 'C': {  
 'confusion\_matrix': [[842, 158],  
 [292, 708]],  
 'accuracy': 0.775  
 },  
 'D': {  
 'confusion\_matrix': [[948, 52],  
 [215, 785]],  
 'accuracy': 0.8665  
 },  
 'E': {  
 'confusion\_matrix': [[909, 91],  
 [102, 898]],  
 'accuracy': 0.9035  
 },  
 'F': {  
 'confusion\_matrix': [[1000, 0],  
 [0, 1000]],  
 'accuracy': 1.0  
 },  
 'G': {  
 'confusion\_matrix': [[959, 41],  
 [59, 941]],  
 'accuracy': 0.95  
 },  
 'H': {  
 'confusion\_matrix': [[999, 0],  
 [0, 999]],  
 'accuracy': 1.0  
 },  
 'I': {  
 'confusion\_matrix': [[1000, 0],  
 [0, 1000]],  
 'accuracy': 1.0  
 }  
}

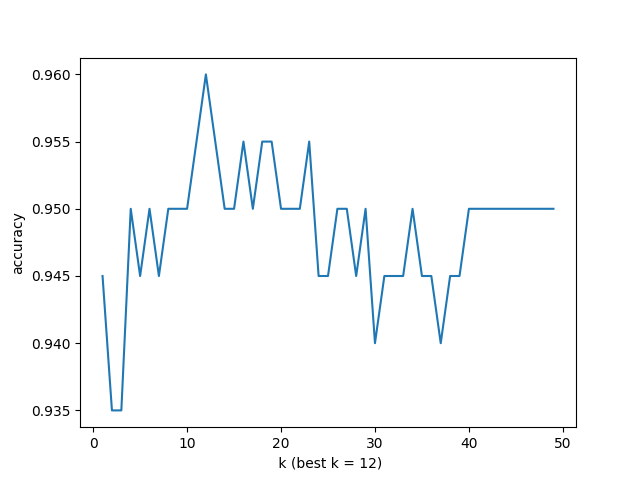
همچنین با استفاده از memory\_profile در پایتون مقدار memory وزمان نیز record شده است (با دستور mprof run classify\_knn.py & mprof plot ) که البته به خاطر سنگین بودن خود این فرایند record زمان بیشتر از حالت عادی است (حالت عادی این الگوریتم حدود 1 ثانیه طول میکشد)

نمودار مصرف حافظه بر اساس زمان در فایل memory\_time واقع در resultآمده است که به صورت زیر است.

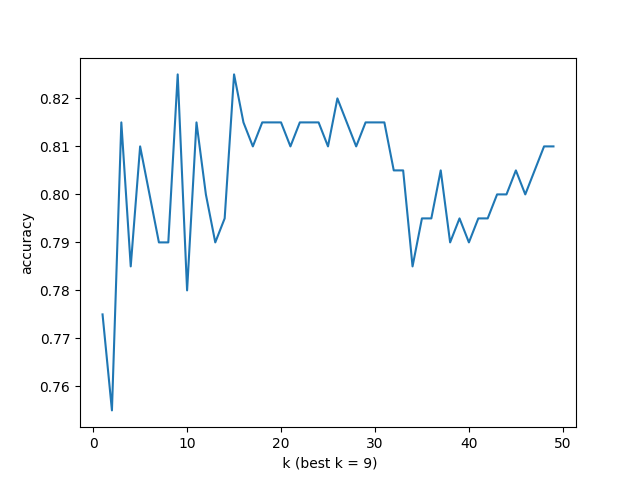


قسمت ب : در فایل best\_k.py عملیات مربوط به این بخش انجام شده است که در آن دقت روی داده‌های validation\_data بر حسب k را به تصویر میکشد که برای هر مسئله ( نه فقط I ) این موضوع انجام شده است و در تمام k های بین 1 تا 200 را به تصویر کشیده است. این تصاویر در فولدر /result/k\_accuracy\_plots آمده است که به شرح زیر نیز است:

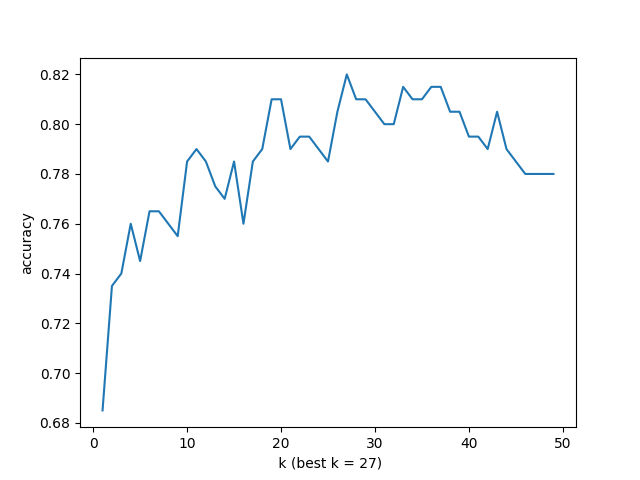
* A



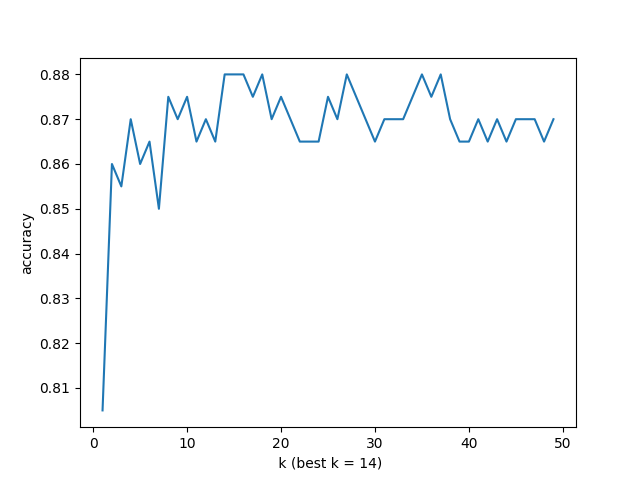
* B



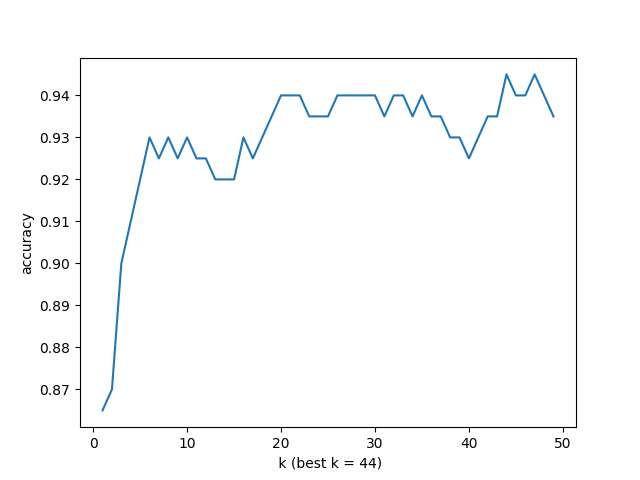
* C



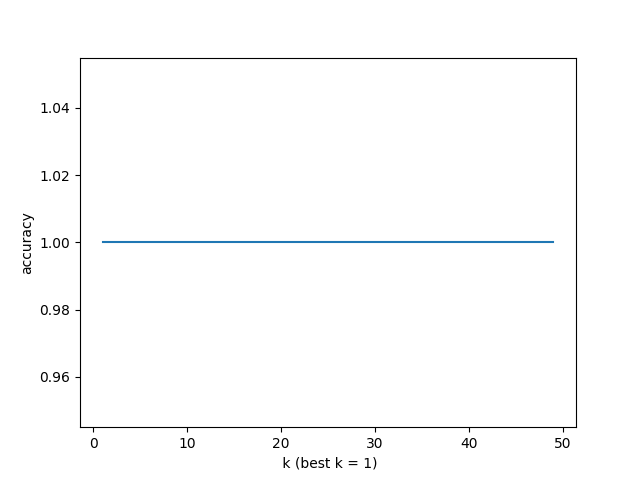
* D



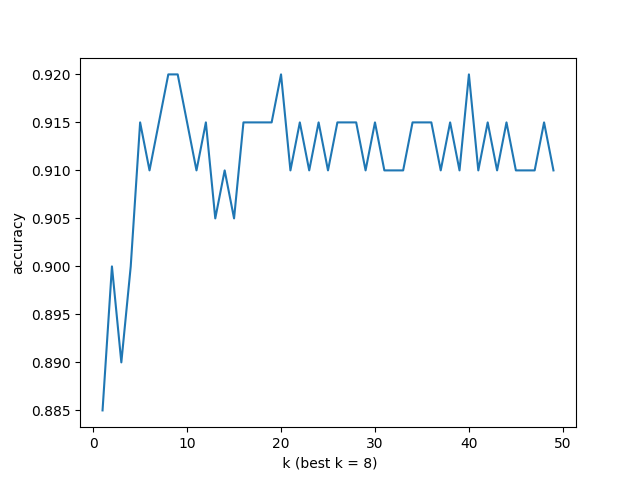
* E



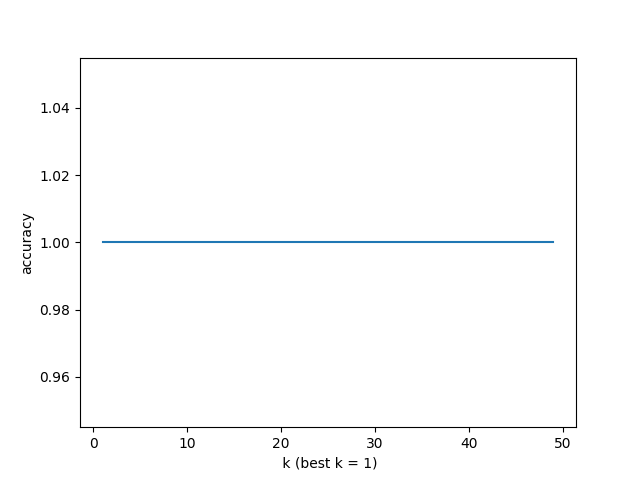
* F



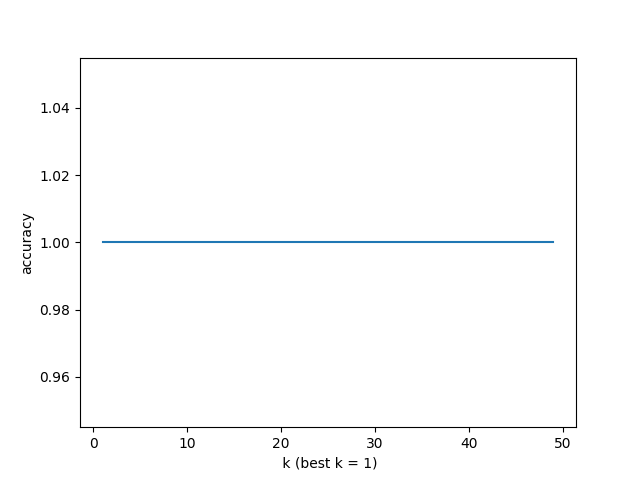
* G



* H



* I



که چون در دسته بندی I داده‌های دو دسته به خوبی از هم تفکیک هستند و فاصله‌ی به نسبت خوبی با هم دارند همواره دقت 100 درصد شده است و k دخیل نبوده است.

در predict هایمان روی داده های تست نیز اثری نداشته و همان نتایج را به ارمغان آورده است.

(بخش 1 را با k=1 تست کردم)

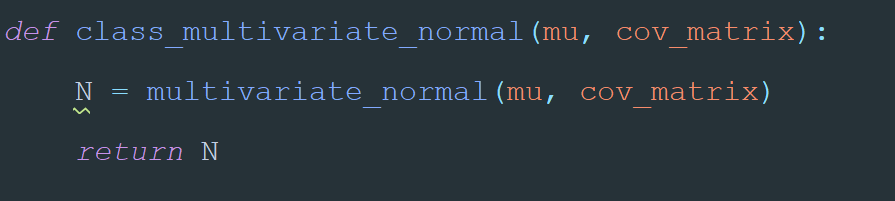
'I': {  
 'confusion\_matrix': [[1000, 0],  
 [0, 1000]],  
 'accuracy': 1.0  
}

## سوال 2

در اسن بخش dataهای استخراج شده از سوال 1 نیز آورده شده است.

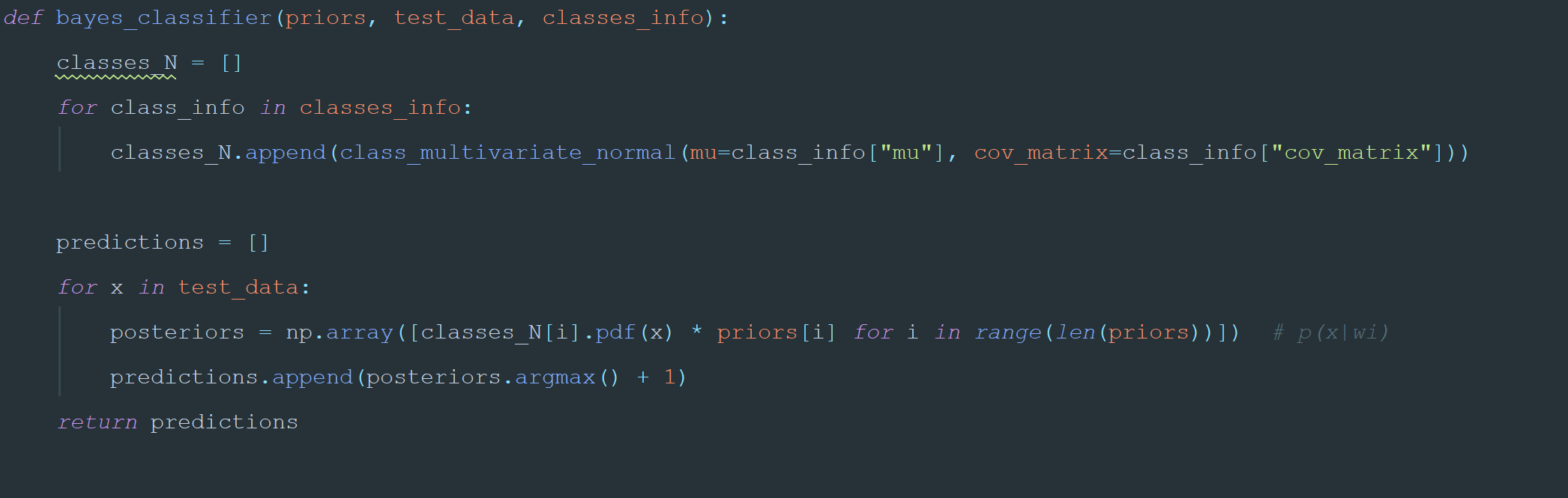
کد های مربوط به این بخش در فولدر problem2 و فایل bayes\_classifier نوشته شده است.

تابع class\_multivariate\_normal یک توزیع نرمال برای هر کلاس متناسب با بردار میانگین و ماتریس کواریانس برمیگرداند



تابع bayes\_classifier بر اساس pdf هر سطر از داده p(wi|x) را حساب میکند و argmaxi آن را حساب میکند و آن داده را به آن کلاس تخصیص میدهد.(تابع بخش اصلی سوال)

(p(wi|x) با استفاده از prior ها وpdf مربوط به x محاسبه میشود)



بخش الف :

تابع مربوط به problem2\_1 مربوط به این بخش است که متناسب با اطلاعات داده شده در جدول داده‌های تست predict شده‌اند.

دسته بندی‌های داده‌های تست در /result/predictions/ مثل سوال قبل آورده شده است.

همچنین دقت و ماتریس تداخل نیز در فایل problem2\_1.json واقع در result نیز ذخیره شده است که به صورت زیر است:

{  
 'A': {  
 'confusion\_matrix': [[980, 20],  
 [71, 929]],  
 'accuracy': 0.9545  
 },  
 'B': {  
 'confusion\_matrix': [[801, 199],  
 [201, 799]],  
 'accuracy': 0.8  
 },  
 'C': {  
 'confusion\_matrix': [[802, 198],  
 [193, 807]],  
 'accuracy': 0.8045  
 },  
 'D': {  
 'confusion\_matrix': [[931, 69],  
 [217, 783]],  
 'accuracy': 0.857  
 },  
 'E': {  
 'confusion\_matrix': [[913, 87],  
 [78, 922]],  
 'accuracy': 0.9175  
 },  
 'F': {  
 'confusion\_matrix': [[1000, 0],  
 [0, 1000]],  
 'accuracy': 1.0  
 }  
}

بخش ب :

کد مربوط به این بخش در تابع problem2\_2 آمده است که مثل همان problem2\_1 است با این تفاوت که مانریس کواریانس قطری است.

نتایج در problem2\_2.json آمده است.

{  
 'A': {  
 'confusion\_matrix': [[1000, 0],  
 [143, 857]],  
 'accuracy': 0.9285  
 },  
 'B': {  
 'confusion\_matrix': [[1000, 0],  
 [1000, 0]],  
 'accuracy': 0.5  
 },  
 'C': {  
 'confusion\_matrix': [[802, 198],  
 [193, 807]],  
 'accuracy': 0.8045  
 },  
 'D': {  
 'confusion\_matrix': [[931, 69],  
 [217, 783]],  
 'accuracy': 0.857  
 },  
 'E': {  
 'confusion\_matrix': [[825, 175],  
 [168, 832]],  
 'accuracy': 0.8285  
 },  
 'F': {  
 'confusion\_matrix': [[1000, 0],  
 [0, 1000]],  
 'accuracy': 1.0  
 }  
}

بخش ب و پ :

دقت‌ها به صورت زیر است:

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| F | E | D | C | B | A |  |
| 1.00 | 0.90 | 0.86 | 0.77 | 0.79 | 0.95 | KNN |
| 0.95 | 0.80 | 0.80 | 0.85 | 0.91 | 1.00 | BAYES\_NONISOTROPIC |
| 0.92 | 0.50 | 0.80 | 0.85 | 0.82 | 1.00 | BAYES\_ISOTROPIC |

در صورتی که ماتریس ھای کوواریانس ھر یک از مجموعه داده ھا را به صورت قطری فرض کنیم، دقت دسته بندی برای داده ھایی که شکل دایره ای ندارند کاھش می‌یابد. دلیل آن پدیده‌ای به اسم برازش مدلی ایزوتروپیک بر روی داده ھای غیرایزوتروپیک است.

به دلیل داشتن توزیع نرمال گوسین این مسائل و به دلیل بهینه بودن دسته بند بیز برای این مسائل دقت مربوط به دسته بند بیز بیشتر از knn است.

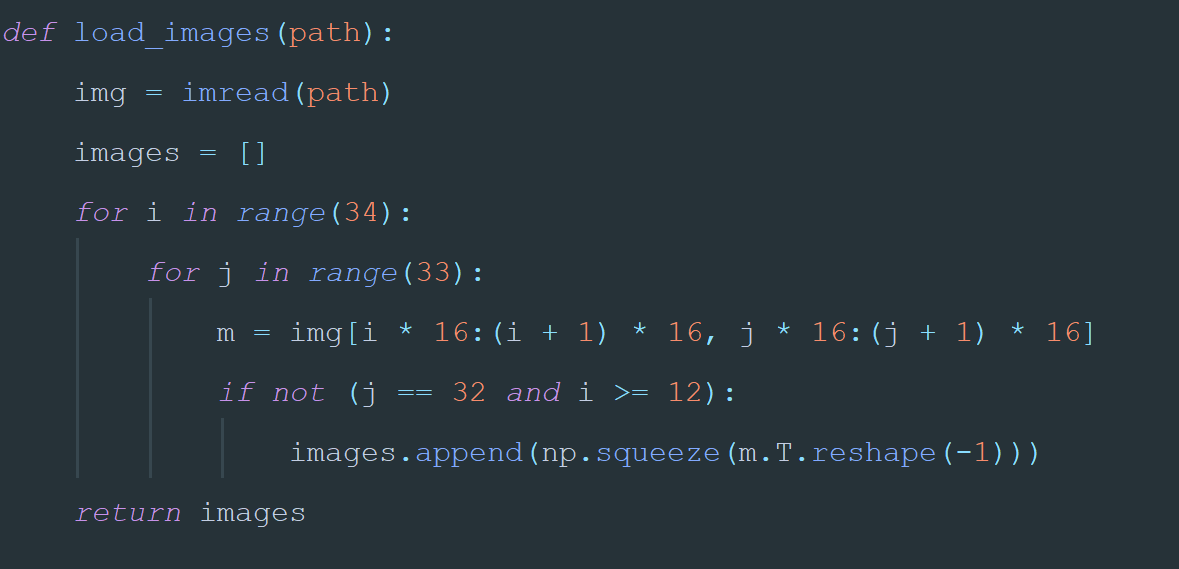
## سوال 3 :

کد‌های مربوط به این سوال در پوشه‌ی problem3 موجود است .

در این پوشه در فولدر images عکس‌های مربوطه موجود است.

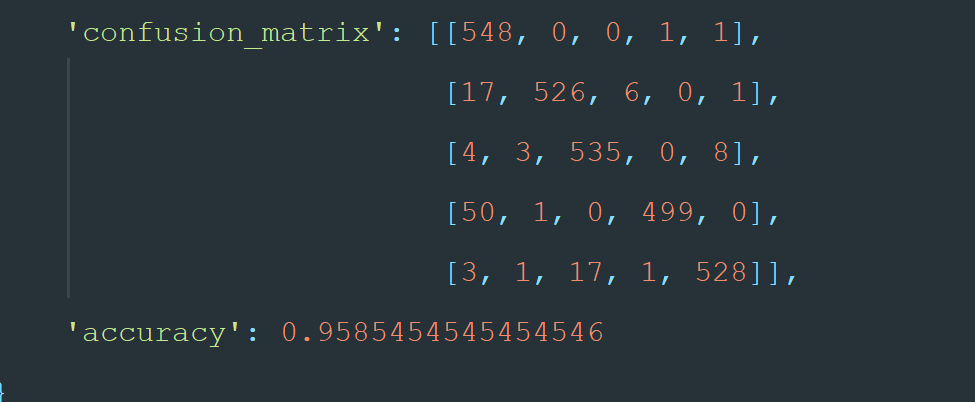
کد‌های مربوطه در فایل problem3.py موجود است.

تابع load\_image وظیفه‌ی خواندن image ها و تبدیل به داده‌ها چندبعدی را دارد.



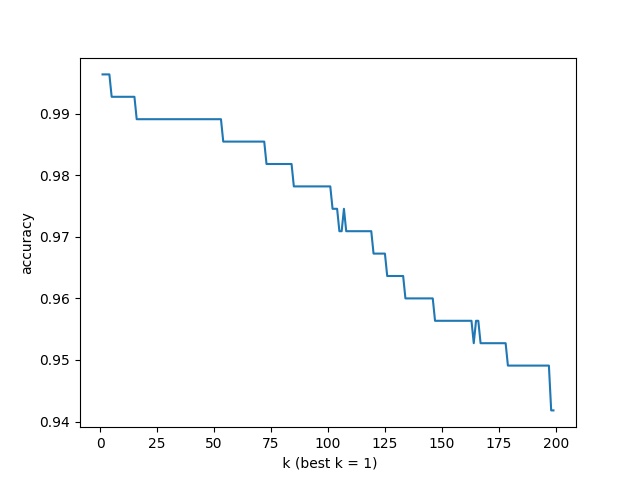
تابع get\_data وظیفه‌ی خواندن عکس‌ها را دارد و به آن‌ها برچسب هر کدام از اعداد را میدهد. همچنین داده‌ها train و test نیز جدا میکند (از هر عدد نصف برای train و نصف برای test ). همچنین داده‌ها کمی به دست آمده در فولدر results/data برای هرعدد ذخیره شده است.

تابع problem3\_1 بخش اول این سوال را انجام میدهد که الگوریتم knn را برای آن اجرا میکند و پیشبینی‌ها را در فایل predictions.csv در فولدر results ذخیره شده است.همچنین ماتریس تداخل و دقت را نیز در فایل accuracy.json در results ذخیره شده است.(با استفاده از توابع سوال 1)



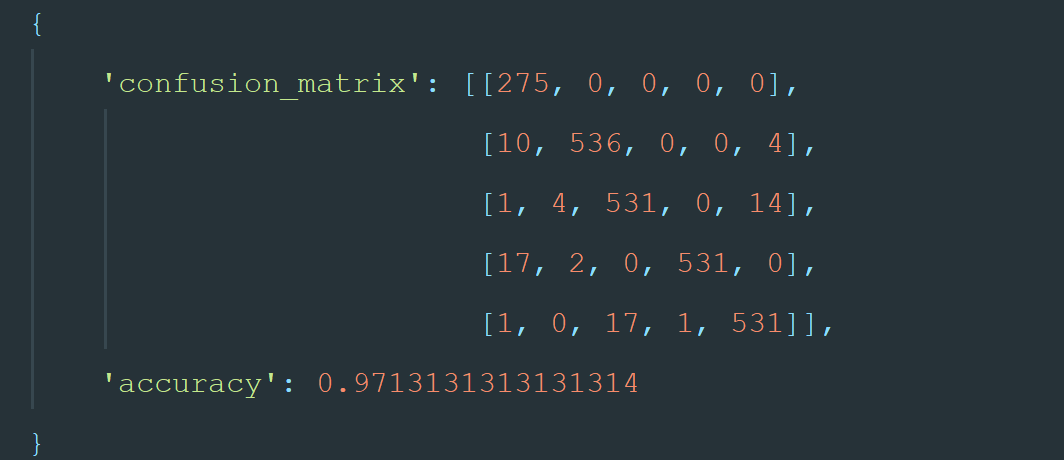
با استفاده از k=10 دقت 95 درصدی حاصل شده است و بیشترین خطایی که وجود داشته به دلیل شباهت 1و4 بوده است و بیتر به دلیل نامناسب بودن اشکال 1 بوده است.

کد مربوط به بخش دوم نیز در تابع problem3\_2 آمده است که دیتاست مربوط به validation\_data جدا میشود و به ازای k های مختلف این بررسی انجام شد که حاصل آن عکس k\_plot.jpg واقع در فولدر results است.



که بهترین k برای 1 (به دلیل خوب تفکیک اعداد) .

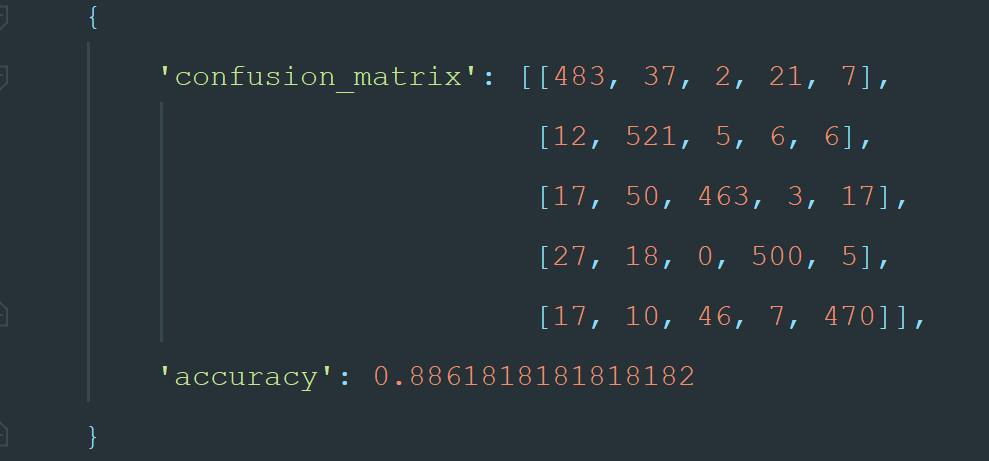
همچنین با k برابر 1 نیز بخش قبل تکرار شد که دقت بالاتری داده است. و نتیجه در accuracy\_with\_best\_k.json ذخیره میشود.



که دقت به 97 درصد رسیده است.

## سوال 4 :

این بخش در فولدر problem4 است که درفولدر images عکس ها موجود است و در فایل problem4.py کد های مربوطه موجود است و با استفاده از تابع GaussianNB عملیات دسته‌بندی با بیز را به طور کامل انجام داده ام و پیشبینی‌ها در فایل prediction.csv واقع در result ذخیره شده است و همچنین دقت و ماتریس تداخل نیز در accuracy.json نیز ذخیره شده است.



که نسبت به knn دقت پایین آمده است (به دلیل نداشتن ساختار گوسین نرمال داده ها)

و بخش ب نیز مثل همان بخش ب ی سوال 2 میباشد.