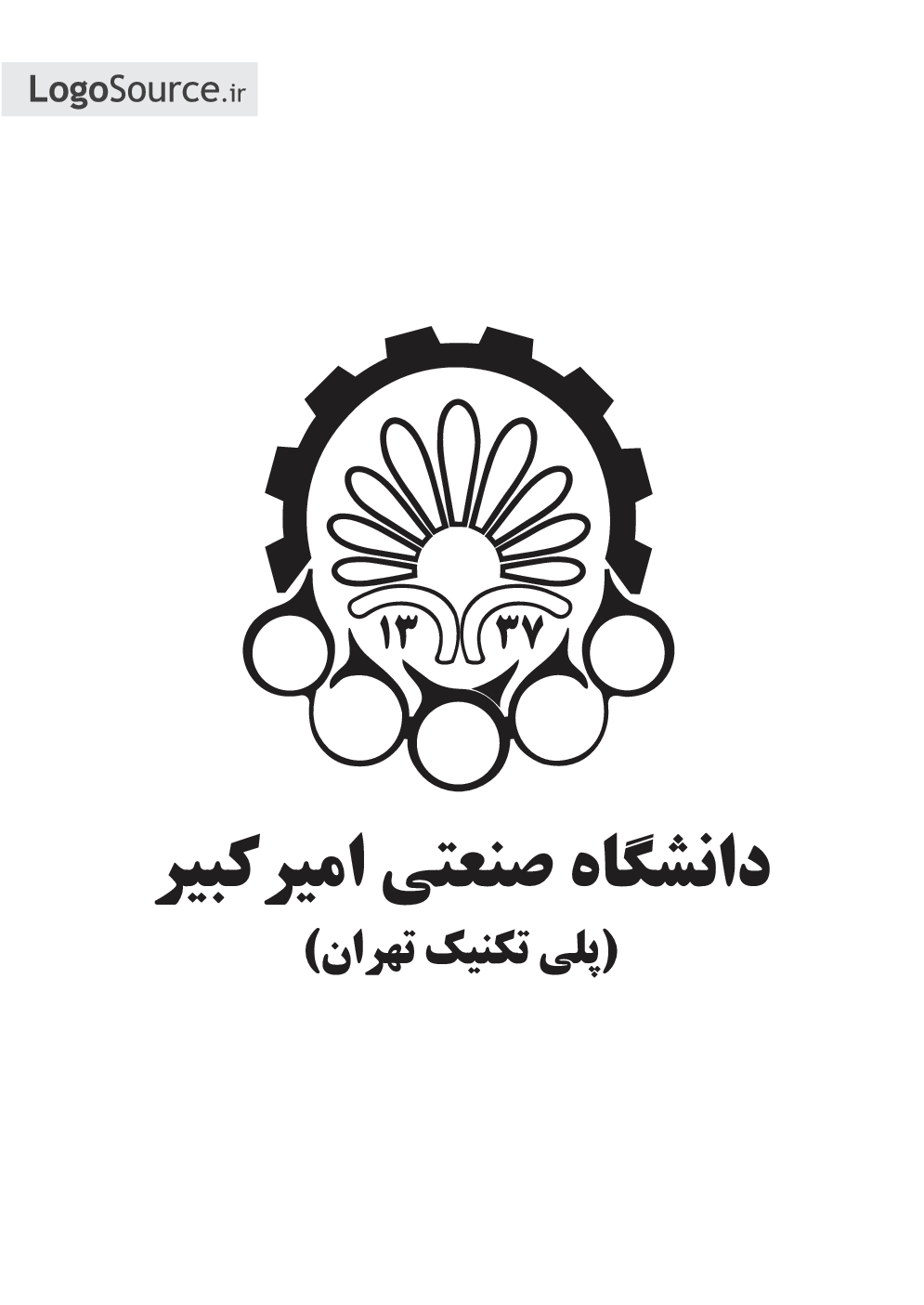
به نام کیمیاگر عالم



**یادگیری ماشین کاربردی**

عنوان

**تمرین سوم (HW03)**

مدرس

**دکتر احسان ناظر فرد**

دانشجو

**امیرحسین بابائیان**

**401131002**

ترم بهار 02-01

دانشکده مهندسی کامپیوتر، دانشگاه صنعتی امیرکبیر (پلی تکنیک تهران)

# فهرست

[فهرست 2](#_Toc134844273)

[سوال اول 5](#_Toc134844274)

[بخش a 5](#_Toc134844275)

[بخش b 5](#_Toc134844276)

[بخش c 5](#_Toc134844277)

[بخش d 6](#_Toc134844278)

[بخش e 6](#_Toc134844279)

[سوال دوم 6](#_Toc134844280)

[بخش a : Pre-processing 7](#_Toc134844281)

[بخش b : Split the data 7](#_Toc134844282)

[بخش c : Run logistic regression 7](#_Toc134844283)

[بخش d : accuracy and confusion matrix 7](#_Toc134844284)

[بخش e : best probability threshold 8](#_Toc134844285)

[بخش f 8](#_Toc134844286)

[ذخیره سازی مدل 8](#_Toc134844287)

[استفاده از مدل ذخیره شده 8](#_Toc134844288)

[سوال سوم 9](#_Toc134844289)

[بخش a 9](#_Toc134844290)

[خواندن داده ها 9](#_Toc134844291)

[حذف ستون بدون کاربرد 9](#_Toc134844292)

[حذف سطرهای خالی کلی و کامنت های خالی 9](#_Toc134844293)

[نصب hazm و nltk روی colab 9](#_Toc134844294)

[مصور سازی داده ها بر اساس لیبل 10](#_Toc134844295)

[عملیات normalizer و lemmatize 10](#_Toc134844296)

[عملیات tokenizer روی متن 10](#_Toc134844297)

[حذف ستون های اضافه 10](#_Toc134844298)

[دانلود stopwords فارسی و حذف کلمات 11](#_Toc134844299)

[لیبل انکودینگ 11](#_Toc134844300)

[تفکیک داده تست و آموزش 11](#_Toc134844301)

[استفاده از CountVectorizer و TFIDFVectorizer 11](#_Toc134844302)

[حذف محتویات اضافه در متن ها 11](#_Toc134844303)

[بخش b 11](#_Toc134844304)

[توضیحات 11](#_Toc134844305)

[پیاده سازی تابع fit 12](#_Toc134844306)

[پیاده سازی تابع predict 12](#_Toc134844307)

[استفاده از GNB 12](#_Toc134844308)

[بخش c 13](#_Toc134844309)

[استفاده از NB داده شده 13](#_Toc134844310)

[GNB در SKlearn 13](#_Toc134844311)

[سوال چهارم 14](#_Toc134844312)

[بخش a 14](#_Toc134844313)

[بخش b 14](#_Toc134844314)

[بخش c 14](#_Toc134844315)

[بخش d 14](#_Toc134844316)

[انتخاب مدل 14](#_Toc134844317)

[اسکیل کردن 15](#_Toc134844318)

[ساخت مدل و ارزیابی اولیه 15](#_Toc134844319)

[پیدا کردن همسایه 15](#_Toc134844320)

[بخش e 15](#_Toc134844321)

[نمونه اول 15](#_Toc134844322)

[نمونه دوم 16](#_Toc134844323)

[نمونه سوم 16](#_Toc134844324)

[نمونه چهارم 16](#_Toc134844325)

[نمونه پنجم 16](#_Toc134844326)

# سوال اول

## بخش a

مدل رگرسیون لجستیک قابلیت پشتیبانی از چند کلاس را به صورت مستقیم دارد، بدون اینکه نیاز به آموزش و ترکیب چندین دسته بند دودویی داشته باشد. این کار با استفاده از تابع softmax در مدل رگرسیون لجستیک امکان پذیر است. تابع softmax برای محاسبه احتمال تعلق هر نمونه به هر یک از کلاس‌ها استفاده می‌شود. در این حالت، تعداد خروجی های مدل برابر با تعداد کلاس هاست و احتمالات تعلق به هر کلاس در خروجی مدل قرار می‌گیرد.

در مدل رگرسیون لجستیک، هدف پیش بینی احتمال برچسب دو کلاس 0 و 1 برای هر نمونه است. اما در مسئله تشخیص چند کلاس، ما باید احتمال تعلق هر نمونه به تمامی کلاس ها را پیش بینی کنیم. برای حل این مسئله، می‌توان از تابع softmax استفاده کرد.

تابع softmax، یک تابع محاسبه احتمالات مخصوص به مدل‌های چند کلاسه است که احتمال تعلق هر نمونه به هر یک از کلاس ها را محاسبه می‌کند. برای مثال، اگر مسئله شامل 3 کلاس باشد، تابع softmax احتمال تعلق هر نمونه به هر یک از این 3 کلاس را محاسبه می‌کند و خروجی مدل 3 عدد احتمالی است که مجموع آن‌ها برابر با 1 است.

در مدل رگرسیون لجستیک با استفاده از تابع softmax، می‌توان تعداد دلخواهی از کلاس ها را پشتیبانی کرد و احتمال تعلق هر نمونه به هر کلاس را برای تمامی کلاس ها به صورت همزمان محاسبه کرد، بدون نیاز به آموزش و ترکیب چندین دسته بند دودویی.

## بخش b

Odds ration یک معیار آماری است که در رگرسیون لجستیک برای مقایسه شانس وقوع یک رویداد بین دو گروه استفاده می شود، به عبارت دیگر نسبت شانس نشان می دهد که احتمال وقوع یک رویداد نسبت به رویداد دیگر به چه شکلی است، اگر بیشتر از 1 باشد یعنی احتمال گروه اول بیشتر است و اگر کمتر از صفر باشد یعنی احتمال گروه دوم بیشتر است و اگر صفر باشد به معنی یکسان بودن شانس است. این نسبت می تواند برای تخمین اثر پیش بینی کننده های مختلف بر نتیجه مورد علاقه در مدل های رگرسیون لجستیک استفاده شود.

## بخش c

در طبقه بندی بیزی، احتمال وقوع یک خروجی یا نتیجه مشخص، با توجه به دسته‌های موجود و مشاهداتی که در هر دسته وجود دارد، محاسبه می‌شود. این روش برای پیش‌بینی و طبقه بندی در بسیاری از حوزه‌ها مفید است، اما معمولاً فقط با داده‌هایی که دارای ویژگی‌های دسته‌ای هستند به خوبی کار می‌کند، به عبارت دیگر، داده‌هایی که متغیرهای کیفیتی (categorical) دارند.

اما اگر داده‌های ما شامل ویژگی‌های عددی هستند، چگونه می‌توانیم از روش طبقه بندی بیزی استفاده کنیم؟ در این صورت، باید ویژگی‌های عددی را به دسته‌هایی تقسیم بندی کنیم. روش معمول برای انجام این کار، استفاده از روش تجزیه و تحلیل بندبند یا binning است. در این روش، با تقسیم داده‌های عددی به چند دسته با ویژگی‌های مشابه، عمل تبدیل دسته‌ای انجام می‌شود که به آن دسته بندی می‌گویند. سپس با استفاده از روش طبقه بندی بیزی می‌توان پیش‌بینی برای دسته‌های مختلف انجام داد.

در مجموع، با تبدیل ویژگی‌های عددی به دسته‌های مشابه و استفاده از روش طبقه بندی بیزی، می‌توان از این روش برای پیش‌بینی و طبقه بندی داده‌های عددی نیز استفاده کرد.

## بخش d

به طور کلی می توان گفت در صورتی که از متد لاپلاس برای حذف صفر شدن احتمال استفاده کنیم، می توانیم احتمال هر حالتی را محاسبه نموده و به نوعی داده ی جدید نیز تولید کنیم، از این رو بله با دیتای کم هم می تواند پاسخ گو باشد.

## بخش e

بله، رگرسیون لجستیک می‌تواند مرز تصمیم غیرخطی را با استفاده از تبدیلات غیرخطی و ویژگی‌های ترکیبی ایجاد کند. به عنوان مثال، با افزودن ویژگی‌هایی که حاصل ضرب دو یا چند ویژگی قبلی هستند، می‌توان مرز تصمیمی غیرخطی را به دست آورد. همچنین با استفاده از تبدیلات غیرخطی مانند تابع لگاریتمی، تابع توانی یا توابع تک‌متغیره غیرخطی، می‌توان مرز تصمیمی غیرخطی را به دست آورد. بنابراین، با استفاده از تبدیلات مناسب، رگرسیون لجستیک می‌تواند مرز تصمیمی غیرخطی را بیان کند.

https://www.linkedin.com/pulse/generating-non-linear-decision-boundaries-using-logistic-d-urso/

# سوال دوم

فایل کد در پوشه ی Source Codes با عنوان P2.ipynb قرارداده شده است.

## بخش a : Pre-processing

تصاویر را در گوگل درایو آپلود نمودم و سپس از گوگل کولب به آن دسترسی ایجاد کردم.



تصاویر را با استفاده از دستورات glob و OpenCV باز کرده و تغییر سایز را اعمال نمودم.



## بخش b : Split the data

تصاویر را به نسبت 25 درصد و 75 درصد به مجموعه تست و آموزش تقسیم کردم.

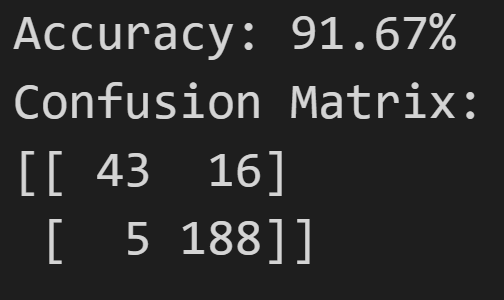


## بخش c : Run logistic regression

فراخوانی رگرسیون لجستیکی از کتابخانه scikitlearn انجام شد و با داده های آموزش، مدل ساخته شد.

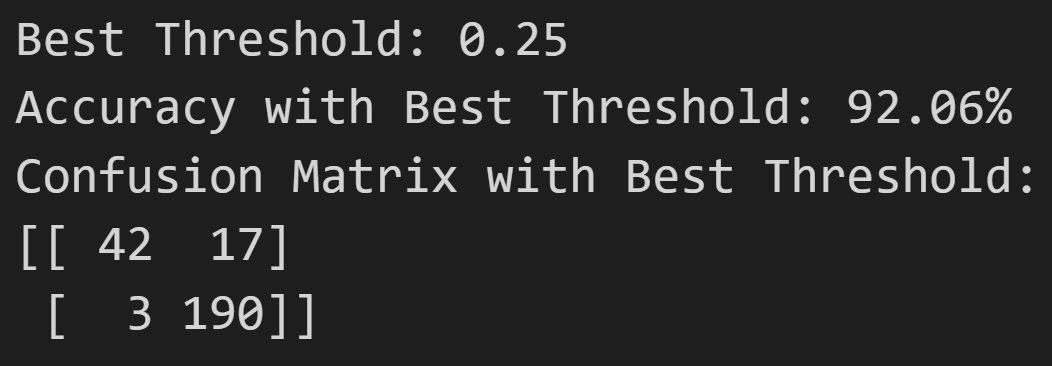
## بخش d : accuracy and confusion matrix

در ابتدا داده های تست را به مدل داده تا پیش بینی نماید و سپس جدول درهم آمیختگی و صحت را چاپ می نماییم.



## بخش e : best probability threshold

معیار f1\_score را با توجه به اینکی مضربی از ویژگی های recall و precision است در نظر گرفته ایم و سپس داده های مختلفی را جهت بدست آوردن بهترین حالت f1\_score یعنی نزدیک شدن به یک را انجام می دهیم، خروجی به صورت ذیل آورده شده است.



بله مقدار کمی بهبود داشتیم.

## بخش f

### ذخیره سازی مدل

با استفاده از کتابخوانه pickle مدل را ذخیره می کنیم.



سپس مجدد مدل را با استفاده از pickle میخوانیم و قدم بعدی را یعنی خواندن مجموعه تست از گوگل کولب و تغییر سایز را اغاز می کنیم.

فایل مربوط به بخش ذخیره ی مدل در پوشه ی Source Codes با عنوان best\_model.pkl قرار داده شده است.

### استفاده از مدل ذخیره شده

خروجی پیش بینی تصاویر را می توانید مشاهده کنید. (در صورتی که کیفیت پایین است، فایل به عنوان ضمیمه نیز ارسال شده است.



فایل مربوط به این تصویر در پوشه ی Report با عنوان P02Result.png قرار داده شده است.

# سوال سوم

فایل کد در پوشه ی Source Codes با عنوان P4.ipynb قرارداده شده است.

## بخش a

### خواندن داده ها



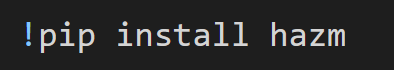
### حذف ستون بدون کاربرد



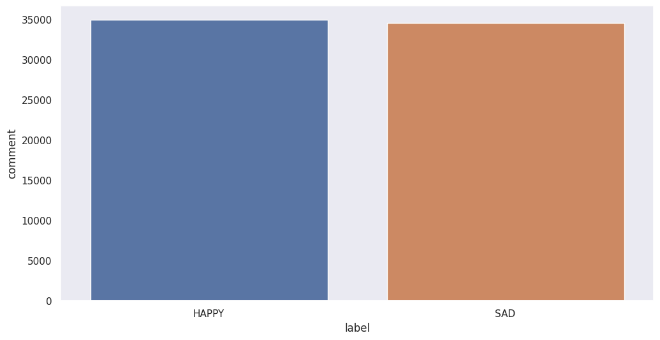
### حذف سطرهای خالی کلی و کامنت های خالی



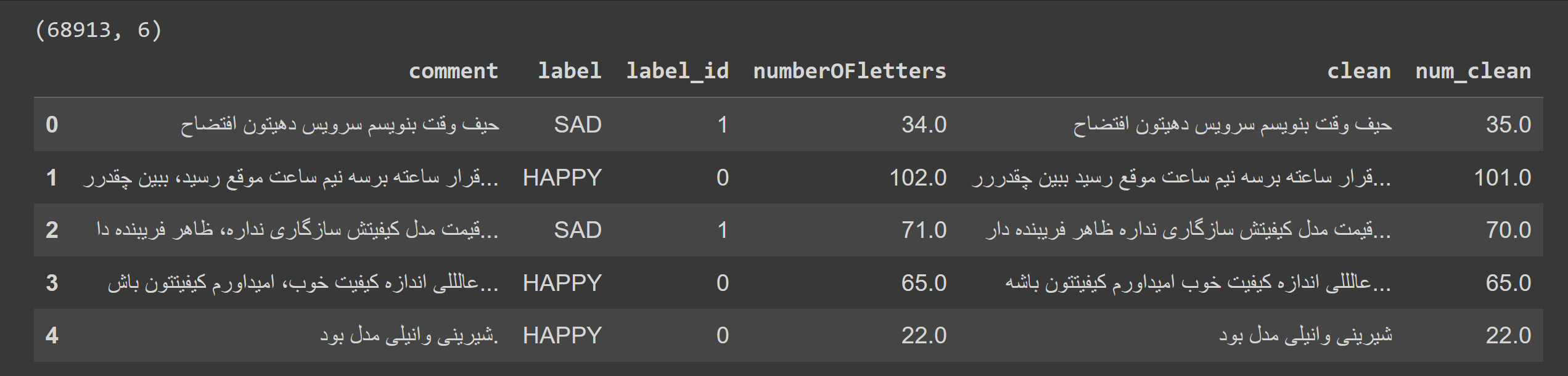
### نصب hazm و nltk روی colab



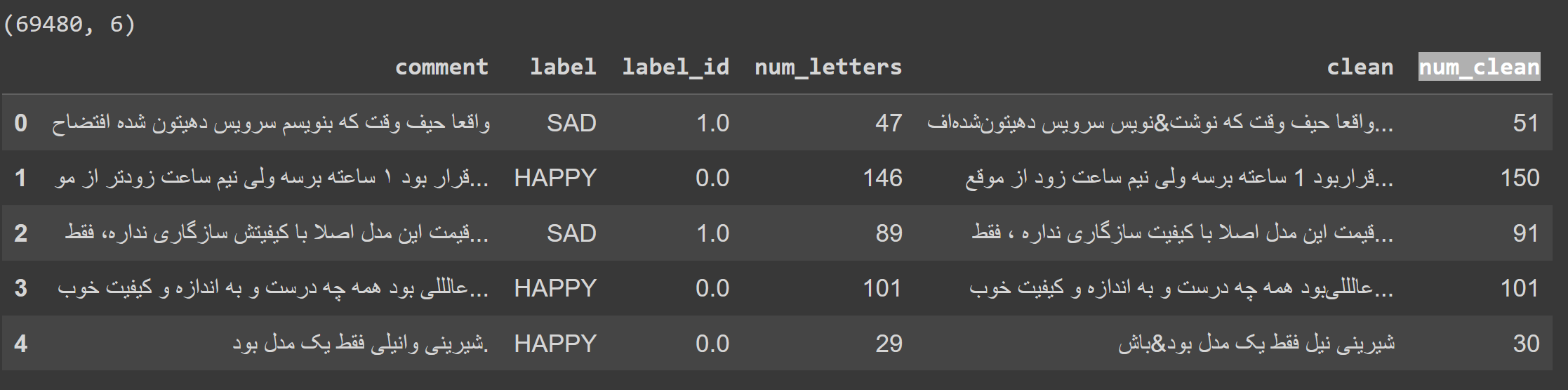
### مصور سازی داده ها بر اساس لیبل



### عملیات normalizer و lemmatize



### عملیات tokenizer روی متن



### حذف ستون های اضافه



### دانلود stopwords فارسی و حذف کلمات



### لیبل انکودینگ

انجام شد.

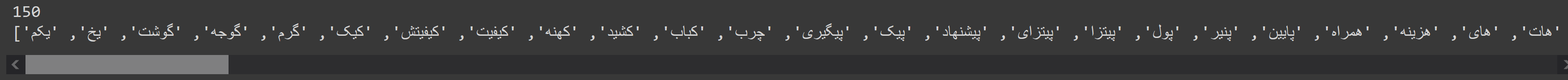
### تفکیک داده تست و آموزش

با نسبت 25 درصد تست و 75 درصد آموزش انجام شد.

### استفاده از CountVectorizer و TFIDFVectorizer

از CountVectorizer برای بدست آوردن 150 کلمه پرتکرار به صورت تک کلمه ای استفاده کردیم و همچنین از TFIDFVectorizer نیز هم.

سپس یک وکتور نهایی ساختیم.



### حذف محتویات اضافه در متن ها



## بخش b

### توضیحات

خب میایم و پس از پیش پردازش داده ها اقدام می کنیم به اینکه کلمات پرتکرار رو به صورت مجموعه داده ها رو بیرون میاریم و لیبل میزنیم و بعدش لیبل رو در هر مجموعه اضافه میکنیم و با این کار یک بردار می سازیم. برای فیت کردن میایم بردار ها رو بر اساس فرمول بندی اساسی NaiveBayes که در لینک آورده شده بود استفاده می کنیم.

برای تست نیز هر جلمه ی ورودی رو اقدام به وکتور سازی ازش می کنیم و در نهایت وکتور رو بررسی نموده و خروجی را مشاهده می کنیم.

### پیاده سازی تابع fit

def fit(x, y, labels):

    n\_label\_items = {}

    log\_label\_priors = {}

    n = len(x)

    grouped\_data = group\_by\_label(x, y, labels)

    for l, data in grouped\_data.items():

        n\_label\_items[l] = len(data)

        log\_label\_priors[l] = math.log(n\_label\_items[l] / n)

    return n\_label\_items, log\_label\_priors

### پیاده سازی تابع predict

def predict(n\_label\_items, vocab, word\_counts, log\_label\_priors, labels, x):

    result = []

    for text in x:

        label\_scores = {l: log\_label\_priors[l] for l in labels}

        words = set(w\_tokenizer.tokenize(text))

        for word in words:

            if word not in vocab: continue

            for l in labels:

                log\_w\_given\_l = laplace\_smoothing(n\_label\_items, vocab, word\_counts, word, l)

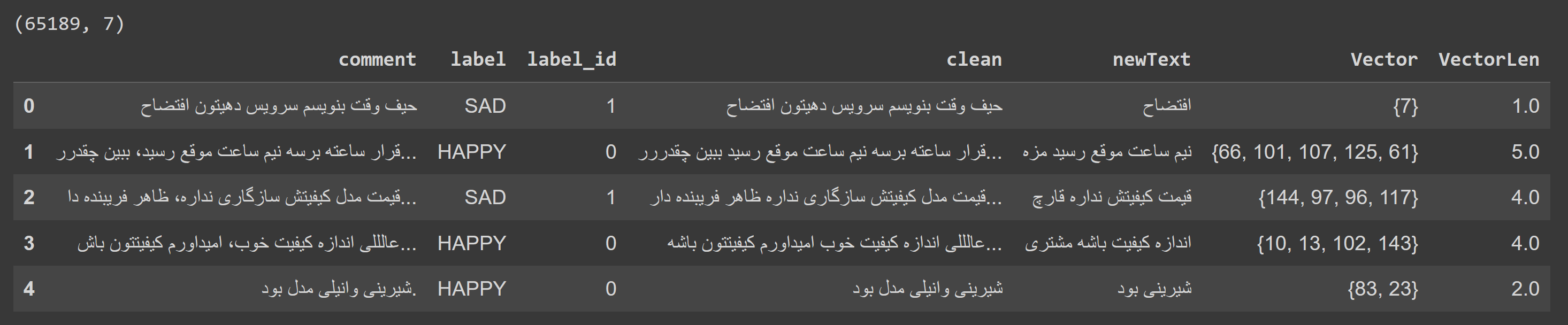
                label\_scores[l] += log\_w\_given\_l

        result.append(max(label\_scores, key=label\_scores.get))

    return result

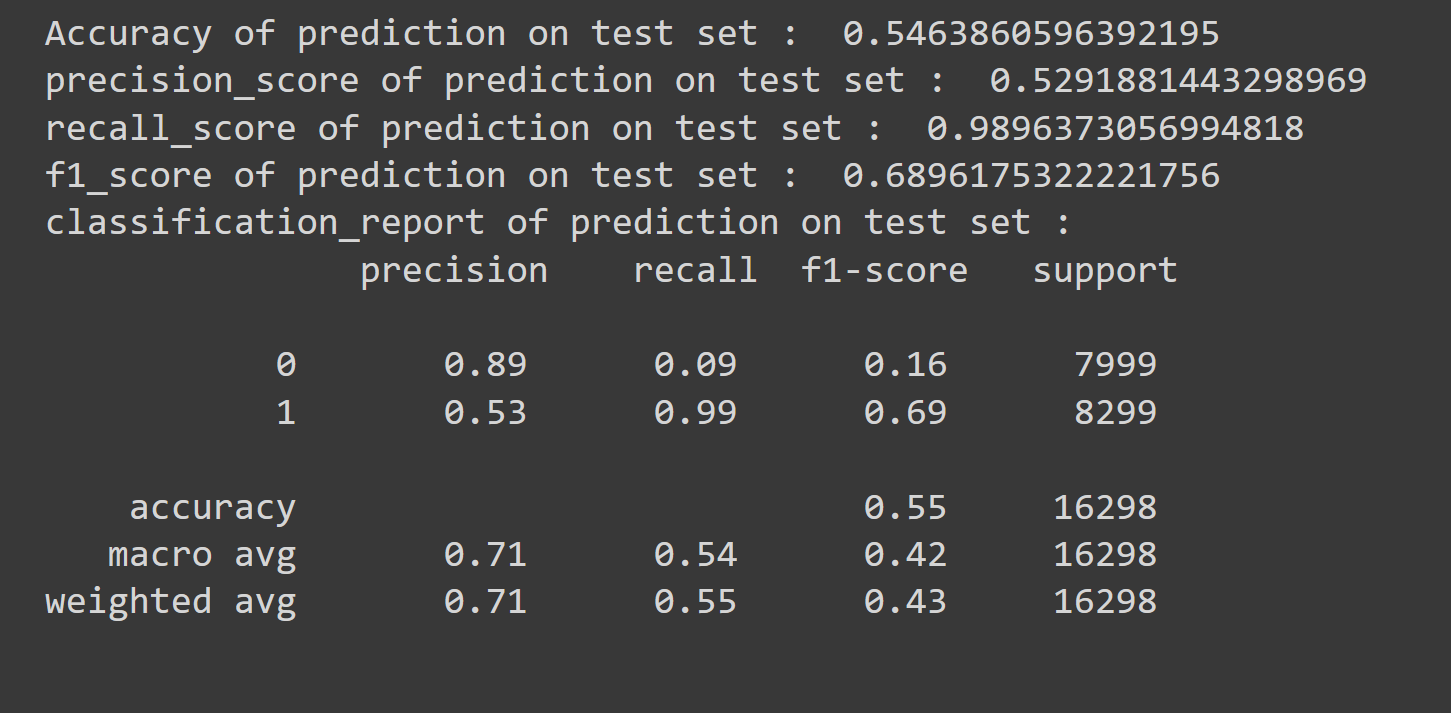
### استفاده از GNB

داده ها را به وکتور تبدیل کردیم طی فرایندی و از خود GNB تعریف شده در sklearn استفاده نمودیم.

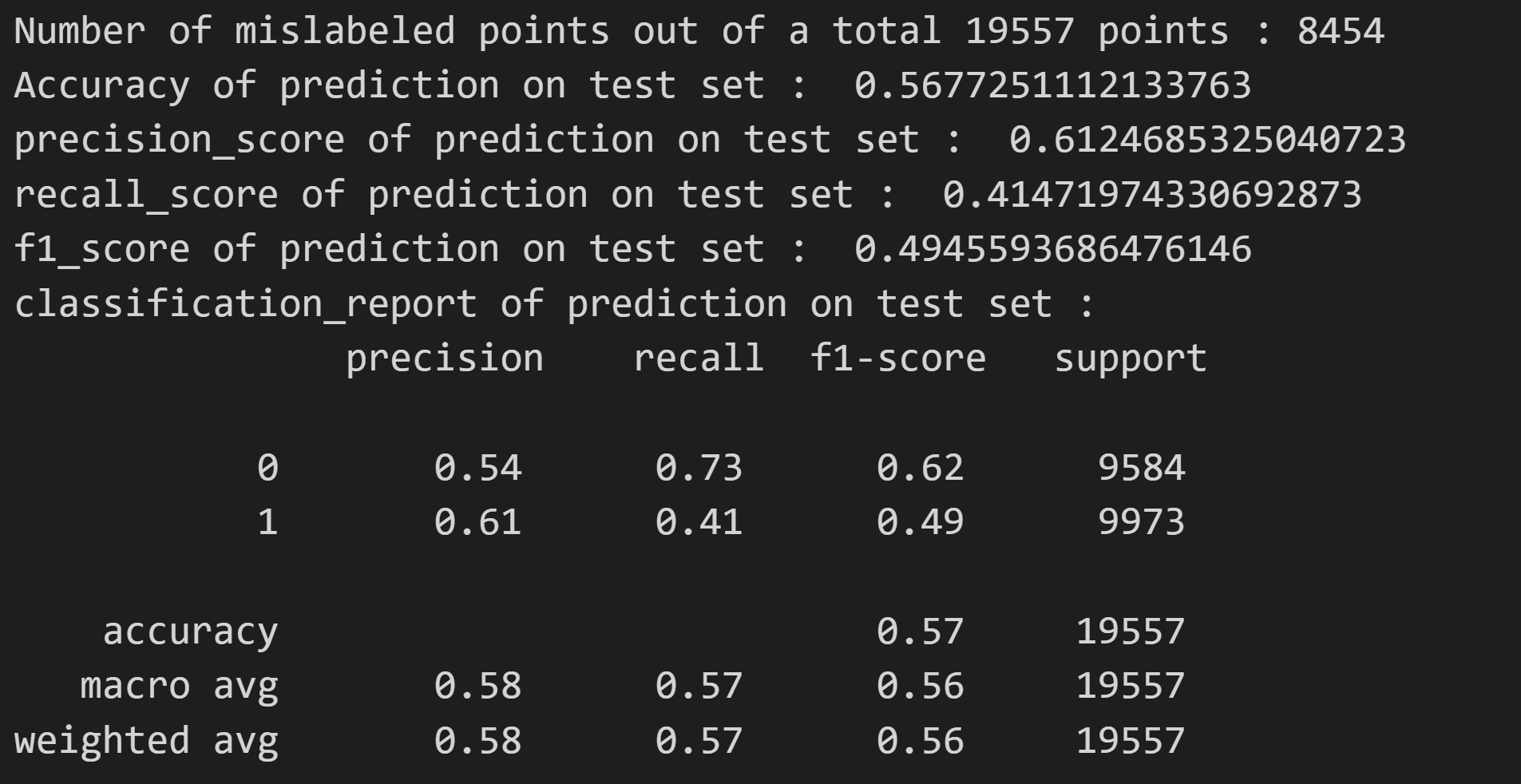


## بخش c

### استفاده از NB داده شده



### GNB در SKlearn



# سوال چهارم

فایل کد در پوشه ی Source Codes با عنوان P4.ipynb قرارداده شده است.

## بخش a

دیتاست لود شد و به دو بخش X و y یا همان تارگت تقسیم شد. سپس با نسبت 0.7 به 0.3 به آموزش و تست تخصیص داده شد.

## بخش b

داده ها را با نوشتن کد نویزی می کنیم، یک نمونه تصویر پس از نویزی شدن :



## بخش c

هدف این است ما بتوانیم نویز را کاهش دهیم، طبیعتا میخواهیم نزدیک ترین تصاویر ممکن را بیابیم و بر اساس آن نزدیک ترین ها که فاقد نویز هستن، تصویر نویزی را رفع نویز کنیم.

## بخش d

### انتخاب مدل

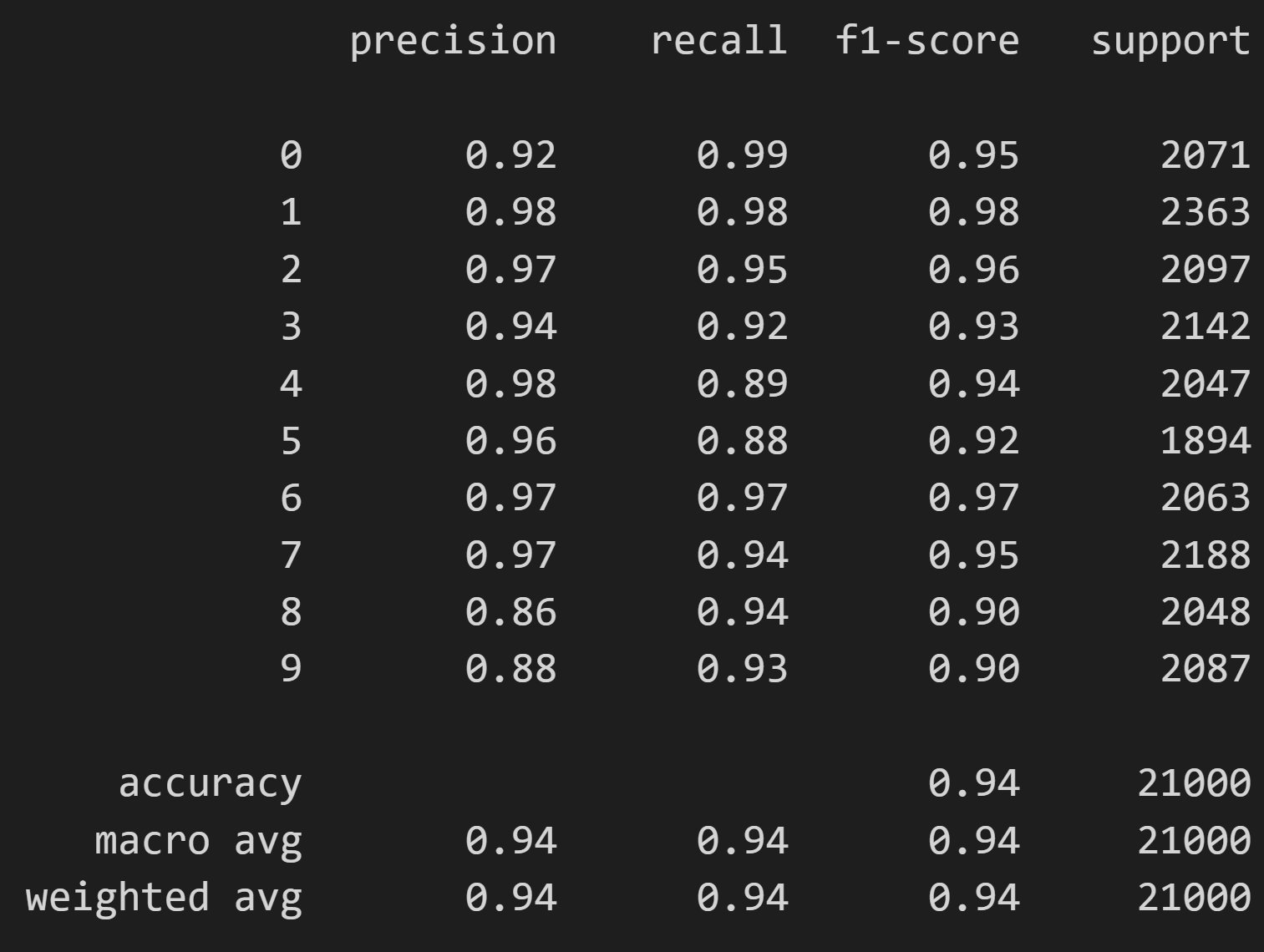
از KNN استفاده کردیم.

### اسکیل کردن

تصاویر را با MinMax اسکیل کردیم تا همیشه صرفا از Standard استفاده نکرده باشیم.

### ساخت مدل و ارزیابی اولیه

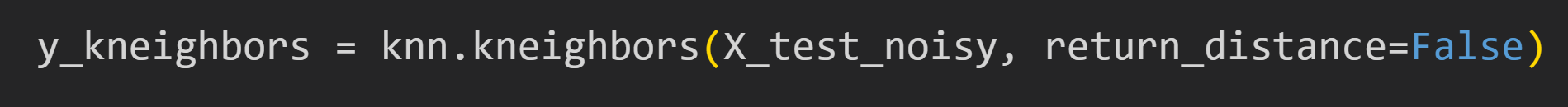
مدل را ساختیم و سپس تصاویر بدون نویز را برای آموزش به طبقه بند دادیم، سپس ارزیابی ذیل را بدست آوردیم.



### پیدا کردن همسایه

در ادامه تصاویر نویزی را برای پیش بینی به مدل دادیم و همسایه های هر کدام از تصاویر نویزی را یافتیم.

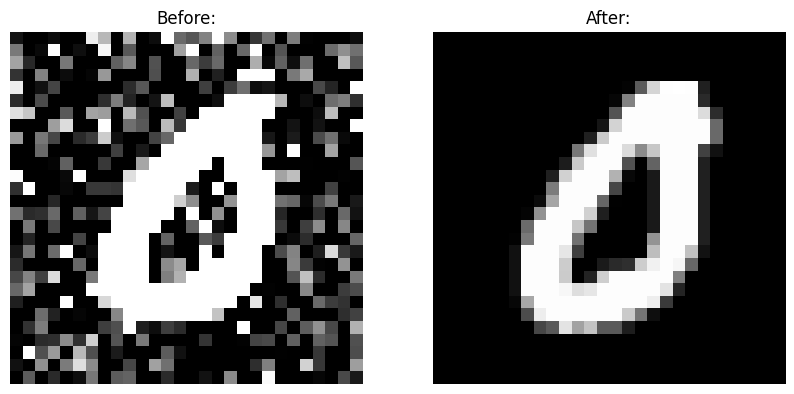
در نهایت از همسایه ها برای بازسازی استفاده کردیم.



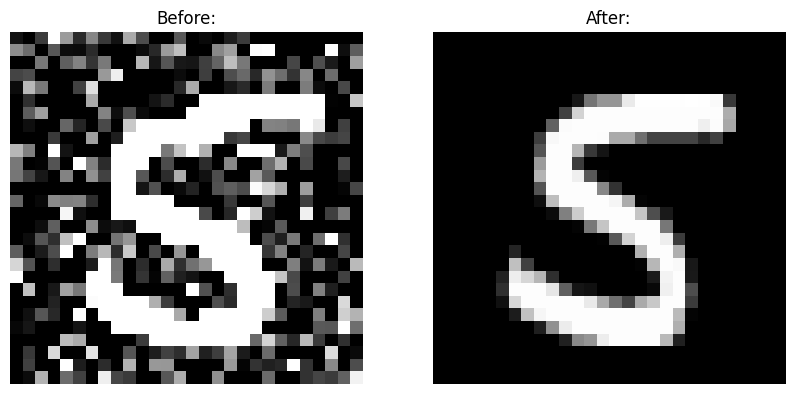
## بخش e

چند نمونه از خروجی کارها درادامه قرارداده شده است:

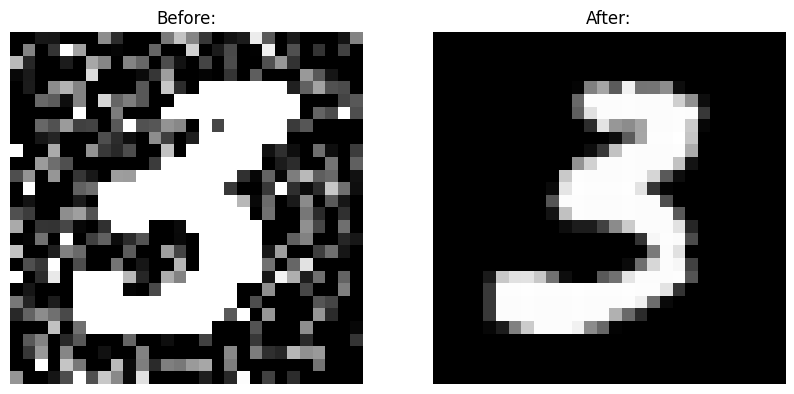
### نمونه اول



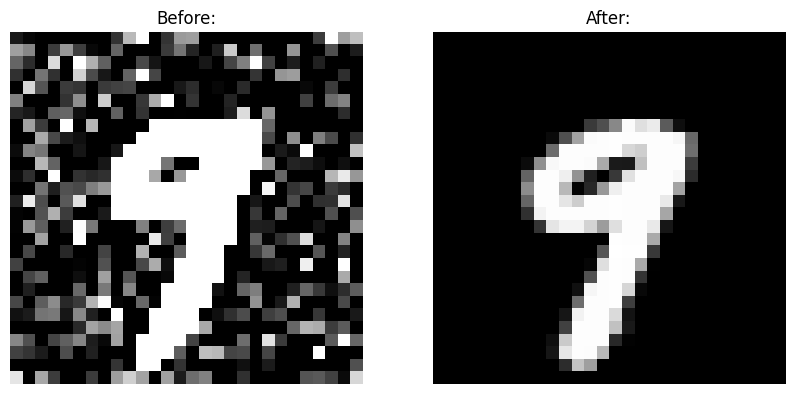
### نمونه دوم



### نمونه سوم



### نمونه چهارم



### نمونه پنجم

اشتباه (3 را 8 کرده است.)

