دانشگاه تهران

پردیس دانشکده های فنی

دانشکده مهندسی برق و کامپیوتر

یادگیری ماشین

تمرین چهارم

امیرحسین عباسکوهی

۸۱۰۱۹۷۵۳۹

استاد ابولقاسمی

سوال ۶)

در این بخش محاسبات با استفاده از کتابخانه numpy و sklearn انجام شده است. فواصل بر حسب فاصله اقلیدسی به دست آماده است.

در بخش اول که تعداد نقاط یعنی تعداد داده ها ثابت است اما تعداد ابعاد تغییر میکند میبینیم که نمیودار فواصل به توزیع uniform نزدیک می شود. knn با تعداد کمی از ویژگی‌ها بهتر عمل می‌کند تا تعداد زیادی از ویژگی‌ها . می‌توان گفت که وقتی تعداد مشخصه‌ها افزایش بیشتری داشته باشد نیاز به اطلاعات بیشتر دارد . افزایش بعد نیز منجر به overfitting می‌شود.

این را در نتایج میبینیم. وقتی فواصل همگی به هم نزدیک شده اند و فرم uniform داریم این نمادی است که knn به خوبی طبقه بندی نخواهد کرد.(در بخش منظور است چون در اینجا برای طبقه بندی خیلی حساس می شود و این یعنی پیچیدگی بیشتر مدل)

knn به علت نفرین ابعاد بسیار حساس به overfitting است . نفرین ابعاد همچنین پدیده‌ای را توصیف می‌کند که در آن فضای ویژگی به طور فزاینده‌ای به خاطر تعداد رو به افزایشی از مجموعه داده‌های آموزشی با اندازه ثابت ، به طور فزاینده‌ای پراکنده می‌شود .

در بخش بعدی هم ما تعداد داده را مورد بررسی قرار دادیم. knn یک الگوریتم دشوار است زیرا مشخص کردن نزدیک‌ترین همسایه یک مجموعه داده بسیار بزرگ می‌تواند پر هزینه باشد. یعنی برای تعداد زیاد این محاسبه خیلی زمان بر است. به همین دلیل زمان بر بودن علی رغم استفاده از محسابات وکتوری باز هم برای تعداد داده بالا خیلی زمان بر است.

سوال ۷)

در این بخش با توجه به cdb بودن فایل داده، خواندن داده را به صورت دستی پیاده سازی کردیم. همچنین در بخش برگرداندن تصاویر یک بخش normalization داریم که می

تواند فعال یا غیر فعال شود. در صورت فعال شدن داده ها به صورت تقسیم بر max نرمالایز می شوند که یک روش معمول برای دیتاست های تصویر است(به خصوص اینکه داده ها همگی بزرگتر مساوی صفر هستند)

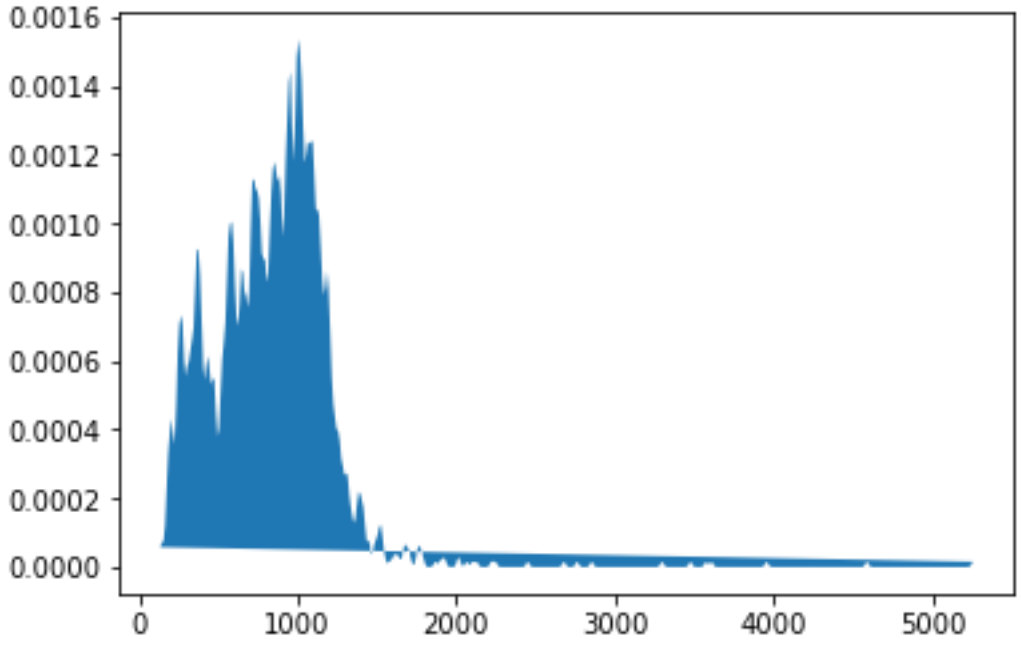
با استفاده از knn زده شده که در آن بر اساس داده ها تمرین فاصله همه نقاط تمرین برای هر نقطه تست حساب می شود و در نهایت در n همسایه نزدیک، کلاس پرتکرار انتخاب می شود، طبقه بندی را انجام می دهیم. نتایج به دست آمده بین knn با sklearn و به صورت دستی کاملا تطابق دارد. همچنین فرقی در بین داده ها با وجود normalization وجود نداشت. این که normalization تاثیری نداشت این است که داده ورودی در اینجا همگی در یک بازه مقدار دارند(۰ تا ۲۵۵) پس اضافه کردنnormalization که به طور کلی برای knn به علت یکسان کردن اثر داده هاست در اینجا اثر خاصی ندارد.

سوال ۸)

نکته: برای این سوال محاسبات در ابتدا با استفاده از توابع آماده حساب شده است سپس با

استفاده از کد دستی.

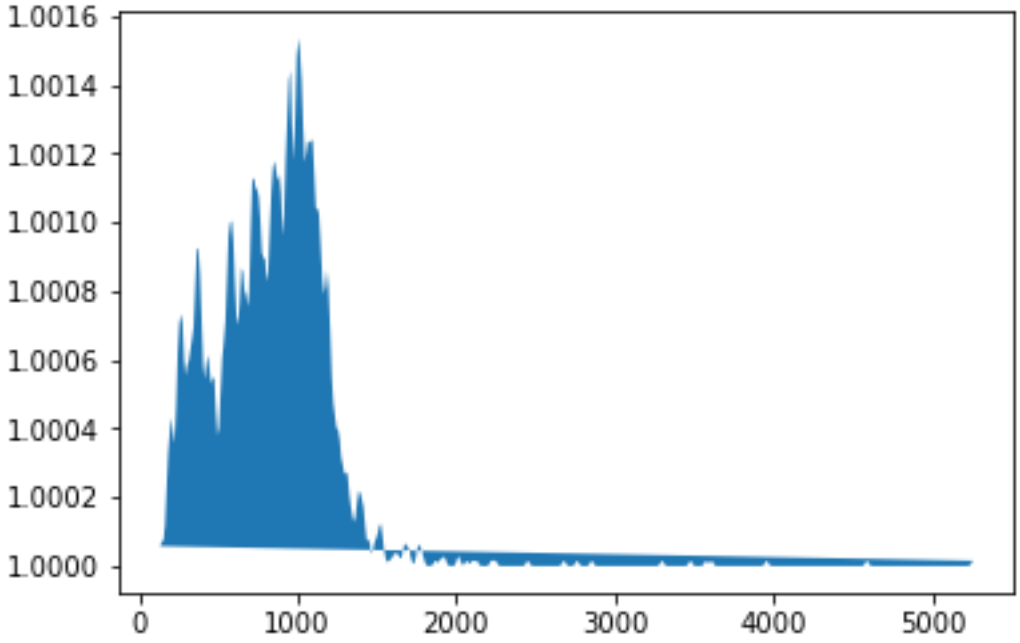
با استفاده از kerneldensity با kernel گوسی از کتابخانه sklearn در ابتدا برای ستون duration توزیع را محسابه میکنیم. سپس یک داده تستی در range داده هایی که قبلا داشتیم(بین min و max) به مدل میدهیم و توزیع را با matplotlib میکشیم که به صورت زیر می شود:

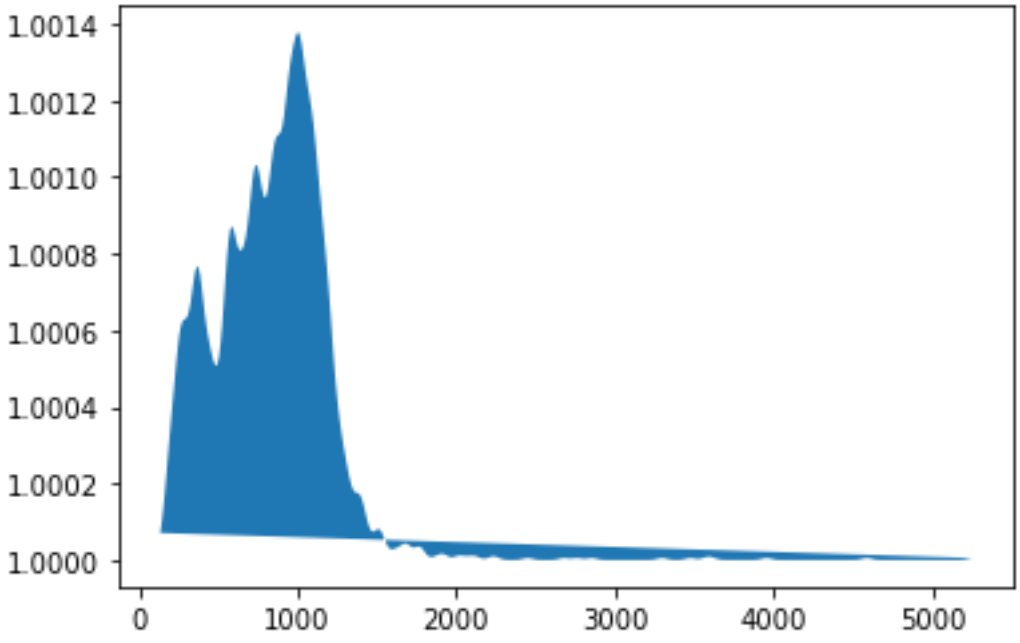


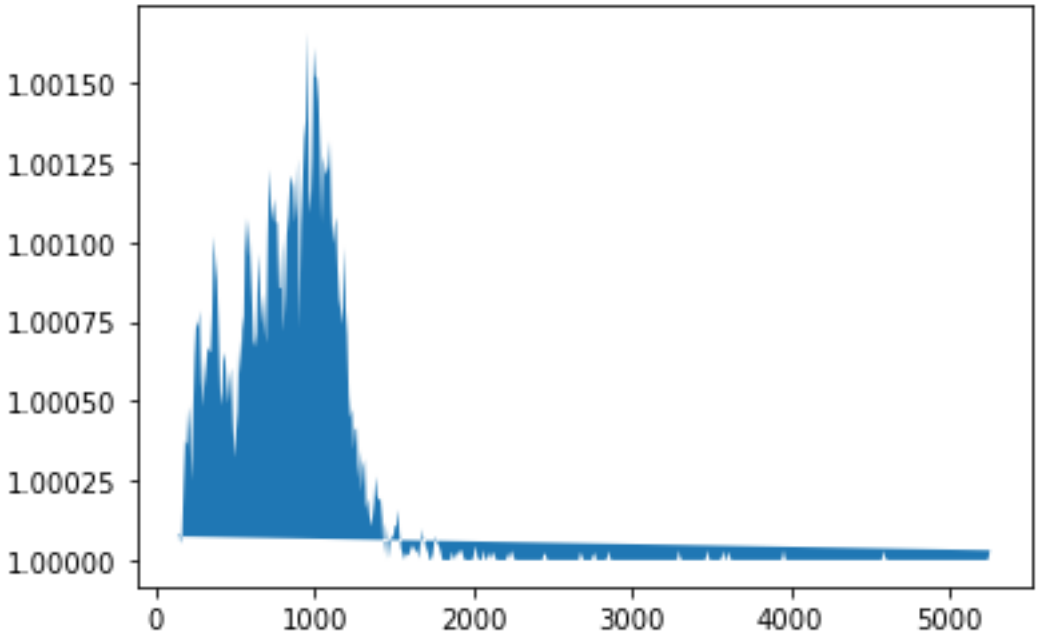
در بخش کد دستی هم بر اساس رابطه پنجره پارزن که به صورت:

$$p_n(\boldsymbol{x}) = \frac{1}{n}\sum_{i=1}^{n}\frac{1}{h^d}\phi\left[\frac{1}{(\sqrt{2\pi})^d h_n^d}\,exp\left[-\frac{1}{2}\left(\frac{\boldsymbol{x}-\boldsymbol{x}_i}{h_n}\right)^2\right]\right]$$

می باشد محاسبات را با numpy انجام میدهیم. تعداد پنجره ها را هم به سه مقدار ۱۰ و ۳۰ و ۵ چک میکنیم.


1.0016
1.0014
1.0012
1.0010
1.0008
1.0006
1.0004
1.0002
1.0000
0
1000
2000
3000
4000
5000


1.0014
1.0012
1.0010
1.0008
1.0006
1.0004
1.0002
1.0000
0
1000
2000
3000
4000
5000



هر چه اندازه پنجره بزرگتر باشد نمودار ساده تر می شود و کمتر به نویز توجه میکند.

در مرحله آخر هم با افزایش n نمودار را می کشیم. در اینجا هم میبینیم با افزایش n مدل به نویز ها حساس و حساس تر می شود.