به نام خدا

نویسندگان : رضا برزگر طرقبه – محمدرضا پوررضا

پروژه نهایی بخش دوم (ژنتیک)

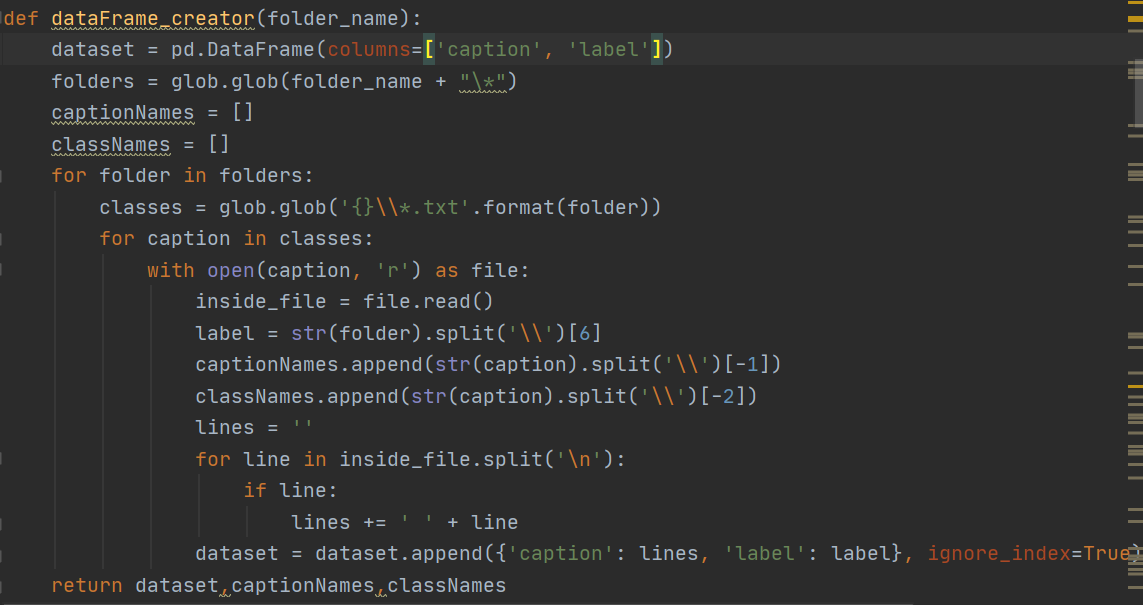
در ابتدای این گزارش به معرفی کتابخانه های استفاده شده برای این پروژه خواهیم پرداخت و استفاده هر کدام را توضیح خواهیم داد.

کتابخانه های استفاده در این پروژه به ترتیت numpy برای کار با آرایه ها و استفاده از توابع مخصوص آنها ، کتابخانه cv2 برای کار با تصاویر ، کتابخانه tensorflow برای کار با مدل داده شده برای encode و decode تصاویر ، کتابخانه pillow باز هم برای کار با تصاویر ، کتابخانه pandas برای کار با داده ها ورودی ، کتابخانه nltk برای پیش پردازش بر روی کپشن های ورودی و در نهایت کتابخانه sklearn برای استفاده از vectorizer آن برای تبدیل کردن کپشن ها به بردار و KNN آن برای پیدا کردن 10 نتیجه نزدیک به کپشن ورودی.

در ادامه لازم به ذکر است از آنجایی که استفاده کردن از کتابخانه ها برای این پروژه بلا مانع بود در این پروژه برای کار با الگوریتم های ژنتیک از توابع آماده ای استفاده شده است که این توابع در فایل ga قرار داده شده اند و از آنها در فایل اصلی یعنی genetic استفاده شده است.

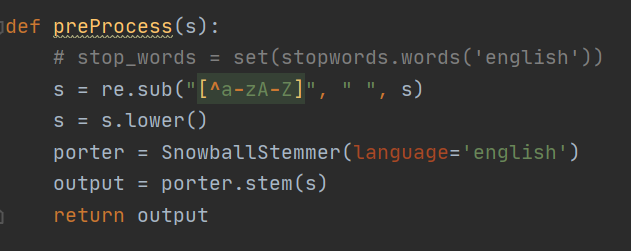
در ابتدای این برنامه ما با استفاده از تابع dataFrame\_creator که پیاده سازی شده است با دریافت یک آدرس دایرکتوری که کپشن ها در آن قرار دارند تمامی کپشن های مربوط به داده های تست و ترین به علاوه اسم هر یک از فایل ها (برای مثال 2008\_02322.txt) و همچنین نام لیبل هر یک از آنها را دریافت خواهیم کرد که از اسم فایل ها و لیبل هر کدام برای پیدا کردن عکس معادل مربوط به هر یک از کپشن ها استفاده شده است.

تابع dataFrame\_creator :

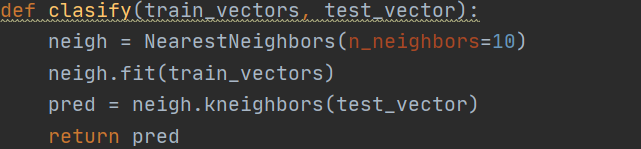


در ادامه بعد از خواندن تمامی داده های ورودی با استفاده از ایجاد ابجکتی از کلاس TFidfVectorizer تمامی کپشن ها ورودی را تبدیل به بردار خواهیم کرد این ابجکت در کانستراکتور خود پارامتری به نام preProcess دارد که مشخص کننده پیش پردازش های انجام شده رو داده ها است که در اینجا ما تابع به نام preprocess تعریف کرده ایم که در آن تمامی کلمات را به حروف کوچک تبدیل خواهیم کرد و سپس با استفاده از snowballstemmer هر یک از کلمات را stem خواهیم کرد.

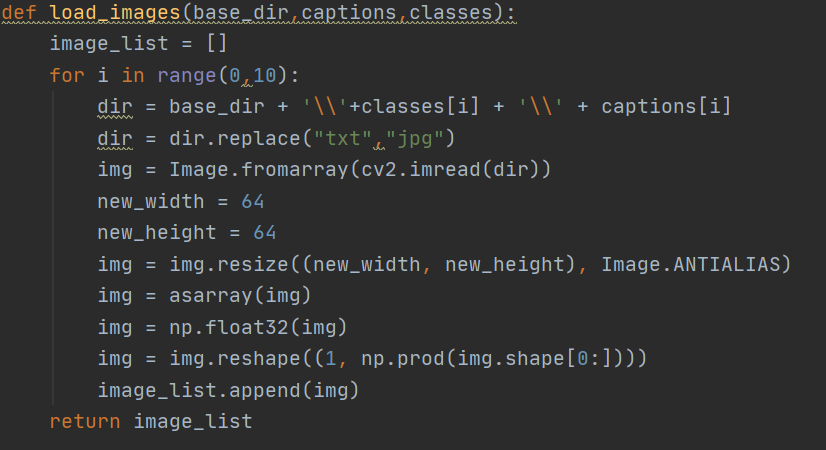
تابع preProcess() :



سپس از کاربر درخواست وارد کردن یک کپشن را خواهیم کرد و سپس کپشن ورودی را نیز با استفاده از ابجکت TFidfVectorizer تبدیل به بردار خواهیم کرد سپس تابعی ایجاد خواهیم کرد که به عنوان پارامتر بردار های کپشن دیتاست و بردار مربوط به ورودی کاربر را دریافت خواهد کرد سپس با استفاده از کلاس KNN 10 همسایه نزدیک به بردار ورودی کاربر را بین بردار های کپشن پیدا خواهد کرد و به عنوان خروجی برخواهد گرداند در ادامه تابع مربوطه قرار داده شده است:



در بخش بعدی بعد از پیدا کردن 10 کپشن نزدیک به کپشن ورودی کاربر، باید عکس های مربوط به هر یک از کپشن ها را از دیتاست خود load کنیم تا تغییرات مربوطه را روی هر کدام از آنها اعمال کنیم که این عمل در غالب یک تابع انجام خواهد شد که به عنوان ورودی 10 کپشن پیدا شده و آدرس دیتاست عکس هارا خواهد گرفت.

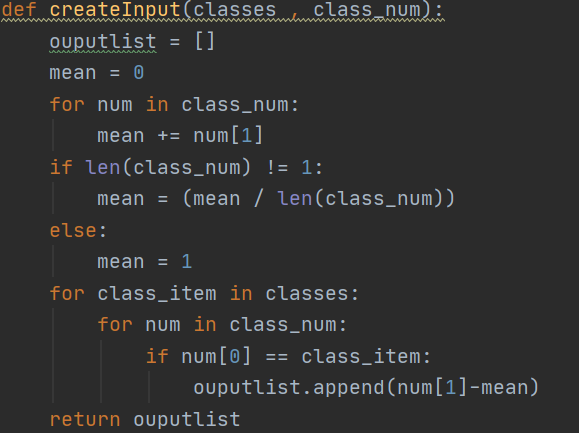


در ادامه با داشتن هر یک از عکس ها با استفاده از توابعی که در صورت پروژه توضیح داده شده بود هر یک از این تصاویر را encode خواهیم کرد.

حال بعد از انجام همه اینکار ها ایده ای که ما برای حل این مسئله استفاده کردیم این است که به ازای لیبل مربوط به هر یک از 10 عکس پیدا شده عددی را به هر یک از این تصاویر نسبت دهیم برای مثال اگر فرض کنیم 10 تصویر پیدا شده شامل 6 تصویر با لیبل سگ و 3 تصویر با لیبل گربه و یک تصویر با لیبل اسب باشد به تصاویر سگ عدد 2.7 را نسبت خواهیم داد ( 3.3 – 6) و به تصاویر گربه عدد -0.3 را نسبت خواهیم داد ( 3.3- 3) و به اسب عدد -2.3 را نسبت خواهیم داد(3.3 – 1) که مقدار 3.3 برابر است با میانگین تعداد تکرار هر یک از لیبل ها با اینکار به دنبال این هستیم که 10 تصویر را به گونه باز نمایی کنیم که لیبل های غالب تاثیر مثبت و لیبل های کمتر تاثیر کمتری داشته باشند در محاسبه وزن و حال برای محاسبه وزن ها فرض میکنیم باید 10 ضریب پیدا کنیم که ضرب هر یک از آنها در ورودی باید ماکسیمم شود پس با اینکار سعی میکنیم برای تصاویر با لیبل غالب ضرایب بزرگ تری پیدا کنیم و برای تصاویر با لیبل که کمتر تکرار شده اند ضریب کوچک تری پیدا خواهد شد. حال بعد از مدلسازی مسئله باید الگوریتم ژنتیک مربوطه را تعریف کنیم که بنا به مسئله تعریف شده ما با 8 کروموزوم که جمعیت اولیه مسئله ما هستند شروع خواهیم کرد و که هر کدام از این کروموزم ها دارای 10 ژن هستند که هر کدام از این ژن ها وزن مربوطه به هر تصویر میباشد. در تابع fitness وزن های ورودی را در 10 تصویر ضرب میکنیم و سپس نتایج به دست آمده را با هم جمع میکنیم تا آرایه بدست آید. سپس آرایه را با استفاده از تابع داده شده در VAE، decode می کنیم.مدل cnn خود را که از قبل آموزش دیده و روی هارد دیسک ذخیره کرده ایم را لود میکنیم. خروجی یک عکس 64 در 64 است اما ورودی مدل ما 380 در 380 است پس با استفاده از تابع resize از کتابخانه pillow آنرا متناسب با ورودی مدل تغییر سایز میدهیم. خروجی مدل یک آرایه است که درصد شباهت ورودی به هریک از کلاس های مربوطه را نگه داری میکنید. حال درصد شباهت کلاس غالب (بین 10 تصویری که بیشترین شباهت را به ورودی کاربر دارند) را از آرایه خروجی مدل انتخاب میکنیم و به عنوان عدد fitness برمیگردانیم.

crossOver: از نوع 1point است ینی هنگام عمل crossOver، دو ژن به دو تکه تقسیم میشوند و جای آنها عوض میشود.

ساخت ارایه ورودی :

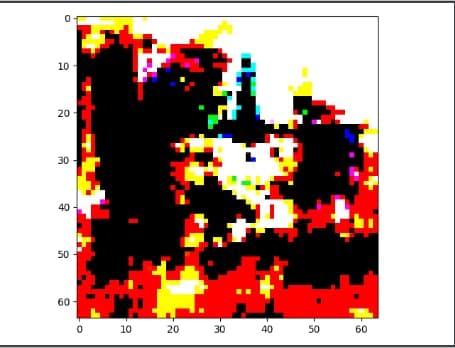


الگوریتم ژنتیک :



سپس با بدست آمدن وزن ها هر یک از وزن های مربوطه را در تصویر مربوط به خود ضرب خواهیم کرد و سپس تمامی این مقادیر انکد شده را باهم جمع میکنیم و خروجی ایجاد شده را decode خواهیم کرد تا تصویر مورد نظر ایجاد شود.

در ادامه به بررسی خروجی به ازای ورودی کاربر برابر با : dog را بررسی خواهیم کرد:



که البته نتیجه نهایی زیاد مطلوب نیست. :) ولی الگوریتم کلسیفایر این تصویر را 36 درصد متعلق به کلاس سگ میداند.