**به نام خدا**

**نام و نام خانوادگی:**

فرناز خوش دوست آزاد

**شماره دانشجویی:**

99521253

**نام استاد:**

دکتر عبدی

**نام درس:**

هوش مصنوعی و سیستم های خبره

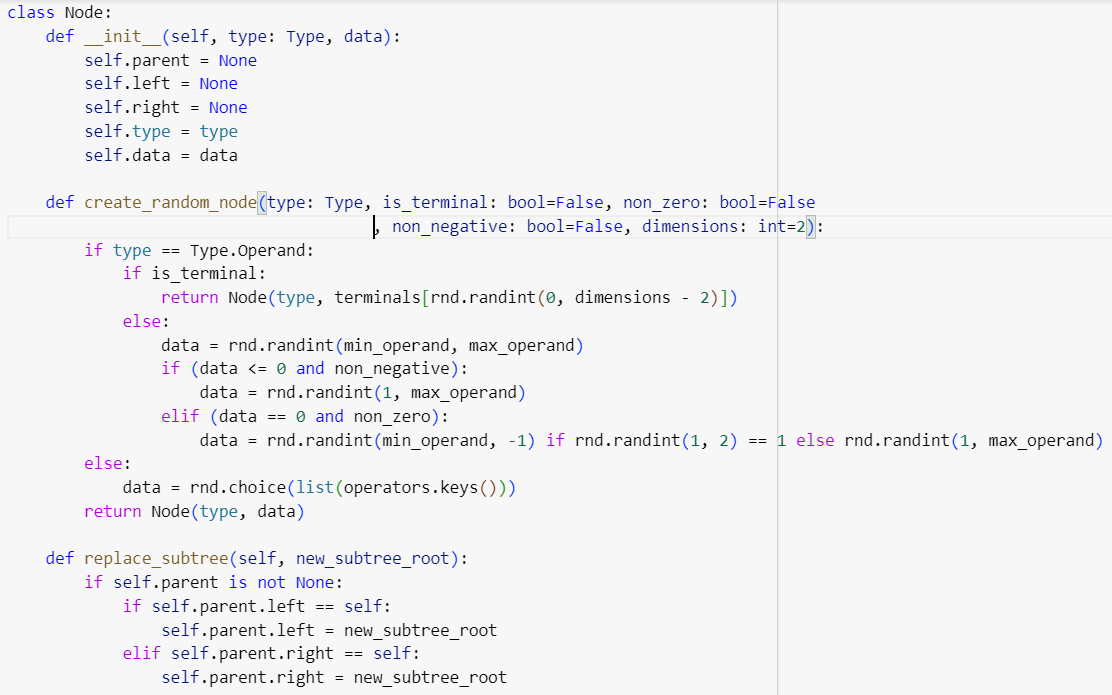
**نام پروژه:**

پروژه ژنتیک

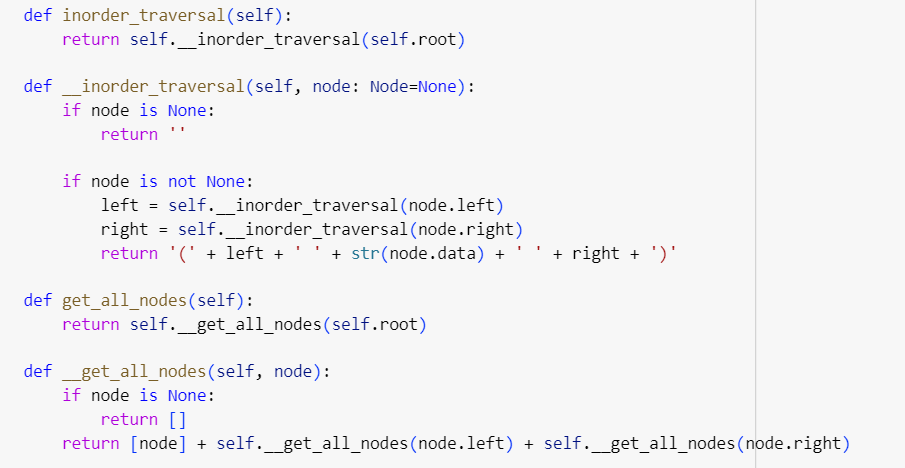
در قسمت اول ما دیکشنری­ای از operators و لیستی از terminals داریم و پس از آن کلاسی از operand و operator داریم تا برای درست کردن tree از آن استفاده کنیم و پس از آن تابعی داریم که از آن برای شمارش operandهای یک operator استفاده می­کنیم که برای sin(x) و cos(x) یک و دیگر عملگرها 2 را برمی­گرداند.

پس از آن برای درست کردن یک tree ما به کلاس node برای نمایش درخت خود استفاده می­کنیم.

در کلاس node ما پنج attributeداریم که از parent و left و right و type و data برخوردار هستند و به غیر از آن دو تابع داریم که برای تولید node رندوم و جابجایی nodeها از آنها استفاده می­کنیم که نام آنها به ترتیب create\_random\_node و replace\_subtree می باشد.

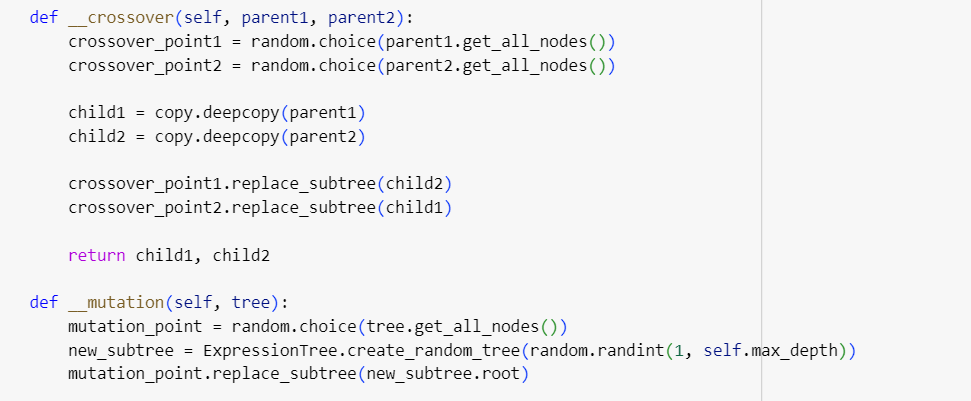


پس از آن از class دیگری به اسم ExpressionTree استفاده می کنیم که تنها attribute آن root می باشد و از آن برای درست کردن tree رندوم و حساب کردن عملیات ریاضی در درخت استفاده می­کنیم که نام این تابع evaluate می­باشد و سپس تابع traversal را داریم که از آن برای traverse در یک tree استفاده می­کنیم و همه ی نودهایleft و right را traverse می­کنیم و تابع get\_all\_nodes() برای گرفتن همه­ی نودهای یک tree استفاده می­شود.



تا اینجای کار تنها کارهای زیر ساختی برای fit کردن درخت را انجام دادیم حال می­خواهیم از کلاس و توابعی برای predict کردن استفاده کنیم که نام این کلاس gplearn می­باشد که دارای 6 attribute می باشد که به ترتیب عبارتند از population\_size و generations و mutation\_rate و population و max\_depth و dimensions.

از population\_size برای تعداد tree های ساخته شده و استفاده می کنیم و آن را به درون population می­ریزیم و dimension در حقیقت همان تعداد terminals قابل استفاده در این کد می باشد و max\_depth حداکثر عمق را در tree به ما نشان می­دهد و mutation\_rate همانطور که از نامش پیداست برای مشخص کردن نرخ جهش از آن استفاده می­کنیم. در این کلاس در ابتدا تابعی به نام \_\_initialize\_generations() استفاده می­کنیم که در آن به تعداد population\_size به درون population list درخت رندوم با استفاده از توابع قبلی که بالا توضیح داده شد اضافه می­کنیم. پس از آن تابعی مربوط به evaluate کردن داریم و از تابع mean\_squared\_error در sklearn برای محاسبه ی خطا و کم کردن خطا در مراحل بعدی استفاده می­کنیم. از cross\_over نیز برای جابجایی در node ها استفاده می­کنیم که به الهام از مباحث زیستی ژنتیک می­باشد. پس از آن تابع mutation را داریم که با توجه به نرخ mutation\_rate() از آن در populatin خود استفاده خواهیم کرد.



پس از آن تابع fit را داریم که از برای fit و predict کردن توابع با استفاده از mean\_squared\_error گفته شده در بالاتر از آن استفاده می­کنیم و پس از آن از توابع مختلف و مشهوری استفاده می­کنیم و به تابع می­دهیم تا آن را محاسبه کند و سپس با استفاده از توابع مختلف آن­ها را نمایش می­دهیم. که همانطور که مشاهده می شود تا حد خوبی توابع درست پیش بینی شده­اند.

در قسمت زیر نمونه­هایی از این توابع نشان داده شده است:

