به نام خدا



گزارش پروژه منطق فازی

درس هوش محاسباتی

دانشگاه اصفهان

دانشکده مهندسی کامپیوتر

استاد درس : دکتر حسین کارشناس

تهیه کنندگان :  
علیرضا دستمالچی ساعی، محمد حسین دهقانی

خرداد 1402

**مقدمه**

در این پروژه خواسته شده بود که از یک دیتاست مخصوص پیام­های ایمیل که مشخص می­کند ایمیل اسپم است یا نه، یک سیستم استنتاج فازی با یک مجموعه قانون بسازیم که با گرفتن ویژگی­های استخراج شده از متن یک ایمیل، اسپم بودن یک ایمیل را تشخیص دهیم.

**پیاده­سازی**

این برنامه به دنبال پیدا کردن بهترین مدل رده‌بندی اسپم و غیر اسپم برای داده‌های متنی است. برای این کار از الگوریتم ژنتیک استفاده شده است.

در این برنامه، ابتدا داده‌های متنی از یک فایل خوانده شده و سپس آن‌ها به یک دیتافریم تبدیل می‌شوند. سپس از روش TF-IDF برای استخراج ویژگی‌های مهم استفاده می‌شود و سپس با استفاده از روش کاهش بعد PCA، تعداد ویژگی‌ها (تعداد بعدها) کاهش داده می‌شود. در ادامه، با استفاده از روش feature\_selection تنها ویژگی­های مهم برای دسته­بندی انتخاب می­شوند.

**الگوریتم تکاملی**

برای الگوریتم تکاملی، از کتابخانه آماده Deap که یک کتابخانه قوی و جامع برای اجرای الگوریتم­های تکاملی است، استفاده کرده­ایم. ابتدا ژنوتایپ یا کروموزوم (Individual) و تابع ارزیابی (Evaluation) را برای toolbox کتابخانه تعریف می­کنیم.

نمایش ما (ژنوتایپ) به صورت یک دیکشنری می­باشد که در toolbox رجیستر شده و دارای 2 بخش می­باشد:

* قسمت Rules که شامل لیستی از قوانین است که به آن Rule base نیز می­گوین، بدین صورت که هر قانون X1 تا Xn و Y دارد. تعداد قوانین هم بین 50 تا 500 عدد می­باشد.
* قسمت Linguistic Functions که برای هر متغیر زبانی یک تابع درجه عضویت تعریف کرده­ایم که در ابتدا به صورت رندوم انتخاب می­شود.

سپس، برای تغییر، از توابعی که خودمان تعریف کرده­ایم استفاده می­کنیم:

* **بازترکیب:**

برای بازترکیب از عملگر تغییر، از one\_point\_crossover استفاده کرده­ایم.

* **جهش:**

برای جهش از عملگر تغییر، با یک احتمالی مقادیر قانون­هارا تغییر می­دهیم سپس با همان احتمال قبلی، توابع عضویت مخصوص هر مقدار زبانی، m و s را جهش می­دهیم.

برای تابع Evaluate که برازندگی هر ژنوتایپ (کروموزم) را حساب می­کند، با استفاده فرمول­هایی که در داک ذکر شده­اند، ساتفاده شده است.

**نتایج**

برنامه برای تعداد جمعیت 10 و تعداد نسل­های 10 با 3 مقدار زبانی، نتیجه زیر را با دقت 90 حاصل کرده است(11 قانون):

Gen nevals avg std min max

0 10 2728.68 2523.3 0 7762.16

1 8 5190.56 1879.14 3164.81 9882.28

2 9 6383.23 1980.43 4108.52 9882.28

3 9 6314.78 2523.94 2983.26 9882.28

4 9 5409.89 3965.59 0 10636.3

5 9 6222.58 3907.63 596.664 12346.7

6 9 7305.13 3014.4 328 9761.59

7 10 5834.59 4636.2 0 14877

8 9 6857.43 4724.04 0 16916

9 9 12228.9 4046.9 4330.8 16923.6

10 8 12447.1 4932.68 4096.39 16923.6

Best individual: {'rules': [[1, 0, 2, 1, 0, 0], [2, 0, 1, 2, 1, 1], [2, 2, 2, 1, 0, 1], [2, 0, 0, 0, 0, 0], [0, 0, 1, 2, 1, 1], [1, 0, 1, 2, 1, 1], [0, 2, 2, 2, 1, 1], [0, 1, 2, 2, 1, 1], [2, 0, 1, 2, 1, 1], [0, 0, 0, 2, 0, 1], [2, 0, 2, 0, 1, 0]], 'ling\_funcs': {0: (<function rect\_trap at 0x7f7b6cd1b400>, 0.2597256470909224, 0.24841580892219772), 1: (<function rect\_trap at 0x7f7b6cd1b400>, -0.053399118156442604, 0.9552181094331375), 2: (<function gaussian at 0x7f7b6cd1ae60>, -0.4030993382928567, 0.4065217311866878)}}

accuracy 0.9063509149623251

با تحلیل نتایج می­توان به موارد زیر دست یافت:

* علت برخی min هایی که در وسط اجرا صفر شده­اند به دلیل mutation می­باشد.
* الگوریتم تکاملی روند صعودی را طی کرده است بجز در نسل 6 که افت کوچکی داشته اما سپس پیشرفت نسبتا خوبی داشته است.
* بهترین ژنوتایپ ما که در بالا آورده شده است، برای 3 مقدار زبانی با توابع درجه عضویت زیر است:
  + Low: rect\_trap
  + Medium: rect rap
  + High: gaussian

با تحلیل قوانین بدست آمده در بهترین ژنوتایپ با اطلاعات ذکر شده در بالا می­توان فهمید که اکثر ایمیل­هایی که اسپم تشخیض داده­شدند مقادیر زبانی Medium یا High را در ویژگی­ها داشتند به خصوص ویژگی چهارم که به تنهایی در یک قانون باعث اسپم تشخیض داده­شدن آن ایمیل شده است.

در نهایت بدست آوردن همچین نتیجه­ای با تقسیم 50 درصدی آموزش و تست عملکرد خوب الگوریتم تکاملی و ژنوتایپ بدست آمده دارد.

در مرحله­ی دوم با اجرای این کد با تعداد نسل­های 10 و تعداد جمعیت 10 ولی با 5 مقدار زبانی و 30 قانون نتایج زیر حاصل شده­است:

Gen nevals avg std min max

0 10 8831.64 7137.22 0 21031.6

1 7 13882.7 6875.92 891.316 21882.6

2 9 13874.7 10396.3 685.361 34121.3

3 7 13236.3 11459.6 1787.57 34121.3

4 9 15715.8 8394.14 4913.41 34121.3

5 10 15381.1 6592.33 7540.41 29822.3

6 10 18054.3 8689.16 1902.88 28923.3

7 10 24218.1 4993.08 14565.1 29614.1

8 9 20289.8 9869.81 740.027 29614.1

9 10 18864.4 10102.3 0 29614.1

10 10 20078.8 9755.71 1958.4 32511

Best individual: {‘rules’: [[4, 3, 1, 4, 1, 1], [2, 4, 4, 2, 3, 0], [1, 1, 1, 3, 4, 1], [0, 1, 3, 3, 2, 1], [4, 2, 0, 1, 3, 1], [2, 0, 3, 3, 1, 1], [1, 1, 2, 4, 4, 0], [3, 3, 0, 2, 1, 1], [3, 2, 3, 2, 0, 1], [1, 4, 1, 1, 0, 1], [2, 1, 1, 4, 1, 0], [2, 2, 1, 2, 3, 0], [3, 2, 2, 0, 1, 1], [3, 1, 3, 4, 1, 1], [4, 2, 2, 0, 3, 0], [3, 0, 0, 2, 0, 1], [1, 1, 2, 4, 1, 1], [3, 4, 4, 1, 2, 0], [0, 2, 3, 4, 0, 0], [1, 3, 2, 2, 4, 0], [2, 0, 2, 1, 3, 1], [0, 4, 4, 0, 4, 0], [0, 3, 4, 1, 2, 1], [3, 0, 4, 4, 1, 1], [4, 4, 1, 4, 1, 1], [1, 4, 1, 3, 2, 1], [1, 4, 0, 4, 4, 0], [4, 3, 0, 4, 1, 0], [4, 1, 1, 4, 1, 0], [4, 3, 4, 4, 1, 1]], ‘ling\_funcs’: {0: (<function sigmoid at 0x7f7b6cd1ad40>, 0.16739159164262007, 0.325079136985742), 1: (<function rect\_trap at 0x7f7b6cd1b400>, -0.17868170411156825, 0.6728158429298997), 2: (<function rect\_trap at 0x7f7b6cd1b400>, -0.7163853770078319, 0.29962110739064896), 3: (<function rect\_trap at 0x7f7b6cd1b400>, 0.8200094237933757, 0.7496362214724762), 4: (<function sigmoid at 0x7f7b6cd1ad40>, 0.9151647467099906, 0.8687030234852108)}}

accuracy 0.7192825112107624

با تحلیل نتایج بدست آمده، می­توان به موارد زیر رسید:

* ژنوتایپ­ها در الگوریتم به مقداری کاهش برازندگی داشته­اند به طوریکه ماکسیمم 34 هزار بوده ولی بهترین ژنوتایپ برازندگی 32 هزار دارد.
* صفرهای موجود در وسط الگوریتم تکاملی بدلیل mutation می­باشد.
* توابع عضویت برای مقادیر زبانی به صورت زیر می­باشد:
  + Very low: sigmoid
  + Low: rect\_trap
  + Medium: rect\_trap
  + High: rect\_teap
  + Very high: Sigmoid

با افزایش تعداد قوانین و مقادیر زبانی با 10 نسل و تست 40 درصد به دقت 71 رسیدیم که نسبت به حالت قبلی مقداری افت دقت داشته­ایم و ممکن به است دلایل مختلفی داشته باشد، مانند:

* مقداردهی اولیه به صورت رندوم با مقادیر خاصی شروع کرده است
* میزان Exploration با احتمالات داده شده برای این الگوریتم کم بوده
* به دلیل علت قبلی، در نقطه بهینه محلی گیر کرده­ایم.

در مرحله سوم نیز، با اجرای کد به تعداد 10 نسل و تعداد جمعیت 10 ولی با تعداد قوانین 50 و تعداد متغیر­های زبانی 7تا، نتایج زیر حاصل شده­است:

Gen nevals avg std min max

0 10 8647.64 4569.4 3369.8 17920.3

1 10 18172.4 7971.29 4556.83 29442.2

2 10 24970.3 9514.38 6760.4 44509.3

3 6 26287 5022.06 18944.7 36058.8

4 10 29019 11429.2 12630.7 41783.6

5 10 26321.3 17298.2 789.168 53192.8

6 10 28479.5 14005.3 6863.96 50048.8

7 8 32525.9 17258.7 5062.54 52320.3

8 10 45356.9 7693.34 25726.7 52320.3

9 9 42068.1 16809.6 6172.81 55757.2

10 8 34474.9 20993.1 725.634 55795

Best individual: {'rules': [[3, 5, 2, 2, 4, 1], [5, 6, 1, 3, 1, 0], [1, 1, 5, 6, 4, 0], [2, 4, 3, 3, 3, 0], [3, 5, 1, 6, 1, 0], [2, 3, 4, 2, 2, 0], [5, 3, 4, 4, 0, 0], [1, 3, 0, 3, 3, 1], [3, 3, 0, 2, 4, 1], [3, 1, 4, 0, 1, 1], [2, 3, 2, 6, 6, 0], [2, 3, 4, 4, 4, 0], [3, 4, 1, 6, 1, 0], [3, 3, 5, 4, 5, 1], [5, 5, 2, 1, 0, 1], [2, 2, 5, 5, 2, 1], [4, 3, 0, 3, 5, 1], [1, 5, 2, 0, 5, 0], [1, 1, 2, 4, 5, 1], [2, 3, 0, 0, 0, 1], [5, 4, 6, 6, 1, 1], [1, 2, 0, 6, 1, 0], [5, 3, 3, 0, 0, 0], [4, 0, 3, 1, 2, 1], [1, 4, 0, 1, 3, 1], [3, 4, 1, 0, 0, 1], [3, 5, 2, 1, 4, 1], [3, 4, 6, 5, 4, 0], [4, 3, 3, 1, 2, 0], [4, 1, 1, 4, 5, 1], [5, 1, 2, 2, 4, 1], [4, 5, 3, 6, 5, 1], [4, 1, 2, 2, 6, 0], [5, 6, 1, 4, 0, 1], [2, 5, 0, 2, 5, 1], [5, 1, 0, 1, 1, 0], [4, 5, 6, 4, 0, 0], [3, 5, 2, 0, 6, 0], [1, 2, 1, 0, 1, 0], [4, 2, 0, 1, 5, 0], [1, 2, 0, 4, 5, 0], [0, 2, 3, 6, 4, 0], [5, 0, 2, 1, 5, 1], [4, 6, 5, 1, 4, 1], [4, 0, 6, 5, 3, 1], [2, 0, 3, 0, 4, 1], [0, 0, 1, 1, 2, 0], [0, 3, 6, 2, 2, 0], [2, 5, 5, 6, 0, 1], [4, 6, 1, 2, 3, 0]],

'ling\_funcs': {0: (<function sigmoid at 0x7f7b6cd1ad40>, 0.4882531421828613, 0.9252488574391238), 1: (<function sigmoid at 0x7f7b6cd1ad40>, 0.18871791792197468, 0.08519210679339584), 2: (<function gaussian at 0x7f7b6cd1ae60>, 0.053283191970501464, 0.7669474526119026), 3: (<function sigmoid at 0x7f7b6cd1ad40>, 0.867398853556133, 0.6941425929969136), 4: (<function rect\_trap at 0x7f7b6cd1b400>, 0.8327768822356998, 0.6584981422332299), 5: (<function sigmoid at 0x7f7b6cd1ad40>, 0.9729642115566914, 0.12098425766441023), 6: (<function sigmoid at 0x7f7b6cd1ad40>, 0.6309064453410838, 0.8866333122999813)}}

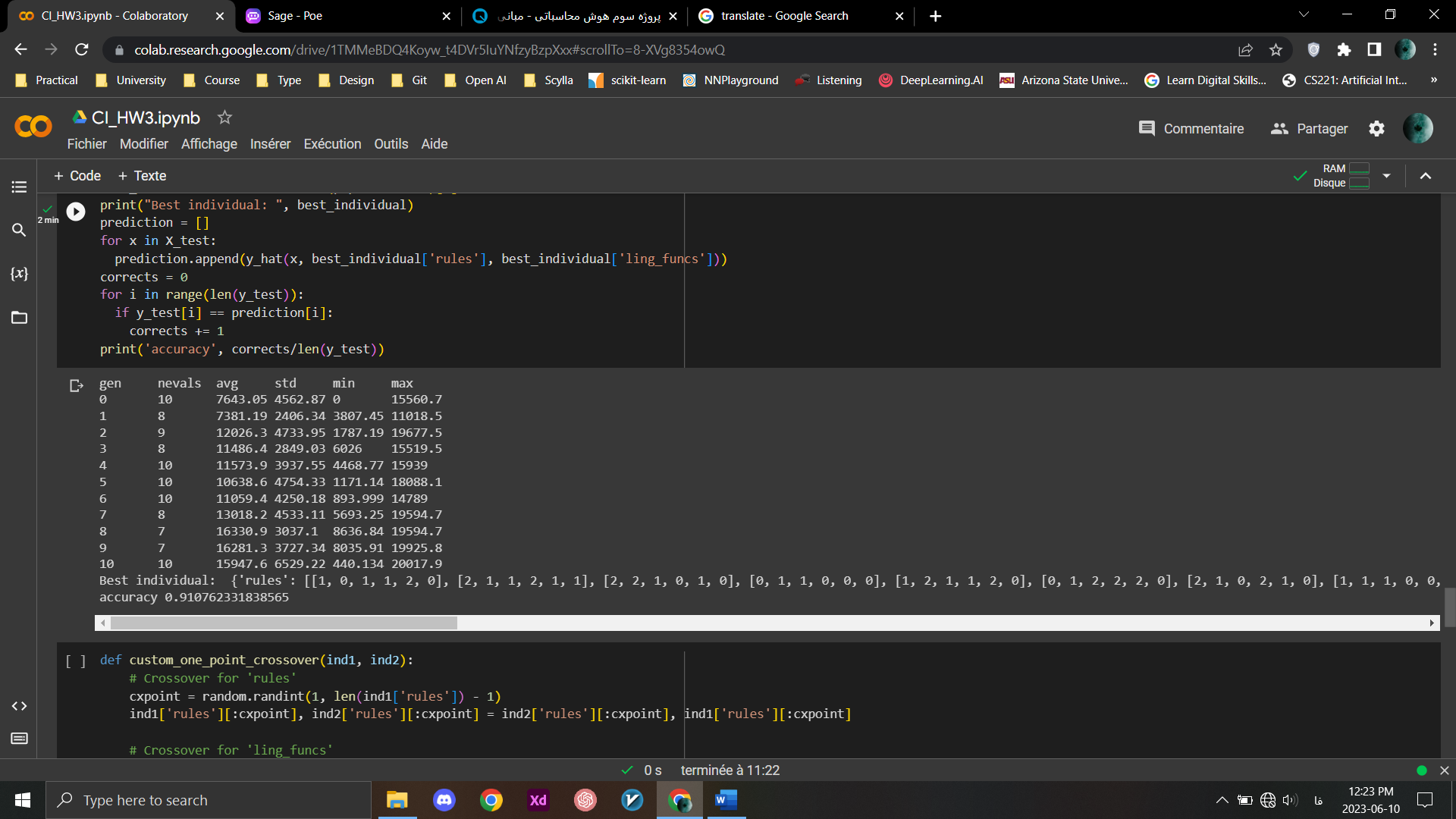
accuracy 0.7726457399103139

با تحلیل نتایج بدست آمده، می­توان به موارد زیر رسید:

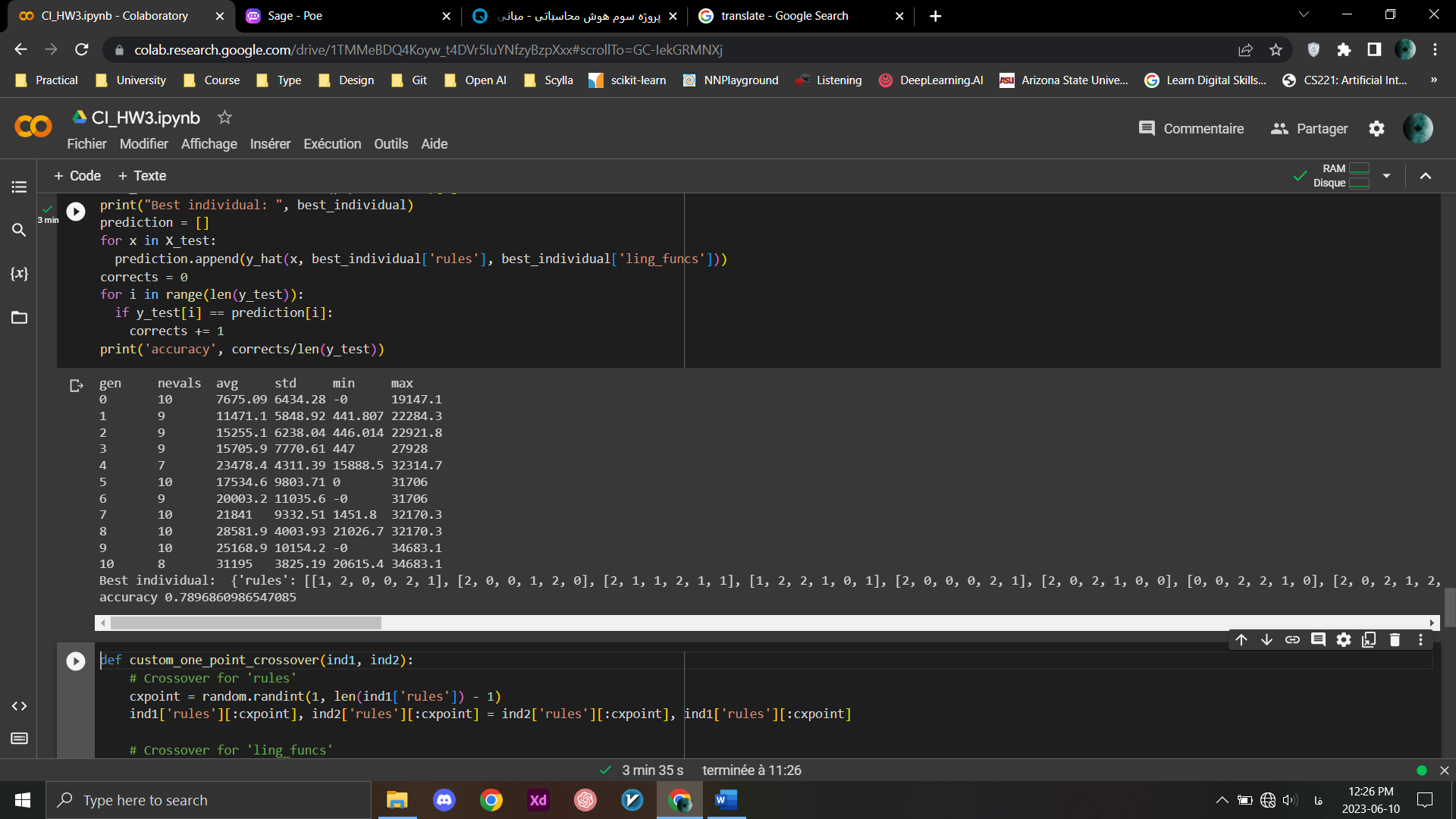
* ژنوتایپ­ها در الگوریتم پسرفت نداشته به صورت افزایشی برازندگی آنها آپدیت می­شد.
* توابع عضویت برای مقادیر زبانی به صورت زیر می­باشد:
  + Very very low: sigmoid
  + Very low: sigmoid
  + Low: gaussian
  + Medium: sigmoid
  + High: rect\_trap
  + Very high: sigmoid
  + High: Sigmoid

با افزایش تعداد قوانین و مقادیر زبانی با 10 نسل و تست 40 درصد به دقت 87 رسیدیم که نسبت به حالت قبلی افزایش دقت 6 درصدی حاصل شد.

یکی از نتایج دیگر با 20 قانون و تنظیمات اجرای قبلی:



یکی از نتایج دیگر با 30 قانون و تنظیمات اجرای قبلی:



استفاده از روش­های کاهش بعد (PCA) و انتخاب ویژگی­ها (feature selection) بستگی به مجموعه داده و مشکل خاصی دارد که به آن پرداخته می شود. استفاده از این تکنیک‌ها در این پروژه باعث شد که عملکرد مدل ما با کاهش تعداد ویژگی‌ها و تمرکز بر مرتبط‌ترین آنها بهبود بخشد و از محاسبات اضافی راحت شدیم. برای بهتر شدن نتیجه­ها شاید می­شد که از روش­های دیگر کاهش بعد مانند MDA و ... نیز استفاده کنیم تا تاثیر دیگر روش­هارا دیده و بهترین نتیجه را انتخاب کنیم.