تشخیص وضعیت بیمار بر اساس توضیحات ارائه‌شده‌ی متنی بیمار

پروژه‌ی نهایی مباحثی در علوم کامپیوتر

دکتر اکبری

پاییز ۹۹

امیرعلی کابلی - ۹۶۱۳۰۲۰

مهراد شاه‌محمدی - ۹۵۱۲۰۳۴

شیده هاشمیان - ۹۶۱۳۴۲۹

صورت مسئله

امروزه با توجه به همه‌گیری استفاده از سایت‌های دارویی که کاربران در آن تجربه و دلیل استفاده از دارویی را با یکدیگر به‌اشتراک می‌گذارند، محتوای متنی تولید شده توسط آن‌ها می‌تواند حاوی اطلاعات مفیدی باشد. این اطلاعات می‌تواند شامل دارویی مصرفی در شرایط خاص، عوارض داروها، محبوبیت یک دارو در بین افراد برای یک بیماری، شرایط بیمارانی که مریضی خاصی را دارند و... باشد. از آن‌جایی که استخراج این اطلاعات می‌تواند در زمینه‌های مختلفی کارآمد باشد از این رو در این پروژه قصد بر این است که با استفاده از مجموعه داده‌ای از پیام‌های نوشته شده توسط کاربران چنین سایت‌هایی، بر روی یافتن حالت و وضعیت (دلیل استفاده نکردن از کلمه‌ی بیماری این است که این حالت می‌تواند سرگیجه یا چیزی شبیه به این باشد) اشخاص با توجه به توضیحاتی که داده‌اند تمرکز داشته باشیم. نکته‌ی قابل توجه این است که این توضیحات افراد در زمان مصرف دارو است.

به بیانی دقیق‌تر هدف در این پروژه آموزش دسته‌بندی است که با گرفتن توضیحات فرد در حال مصرف دارو، شرایطی (یا بیماری‌ای) که شخص در آن قرار دارد را تشخیص دهد. هدف اولیه پروژه تنها بررسی تعدادی از پرتکرارترین وضعیت‌ها (بیماری‌ها) است که این تعداد با بررسی دقیق‌تر داده‌ها انتخاب خواهد شد.

چکیده

ساختار مد نظر ما، آموزش مدلی برای تشخیص بیماری از روی شرح ارائه شده است. برای انجام این کار،

در ابتدا، اطلاعات مربوط به فیلدهای مورد نیاز از مجموعه داده‌ی انتخاب شده را جدا می‌کنیم. در مرحله‌ی بعد، این فیلدها را پیش‌پردازش می‌کنیم. در ادامه به کمک داده‌های پیش‌پردازش شده، با در نظر گرفتن شرح‌های متنی به عنوان داده‌ی ورودی و بیماری‌ها به عنوان برچسب مد نظر، مدل‌های Logistic Regression، SVC، SGD، Decision Tree، Random Forest و Adaboost را آموزش می‌دهیم که با توجه به شرح ورودی، پیش‌بینی‌ای در مورد بیماری‌های ممکن خواهد داشت.

سپس آن‌ها را ارزیابی کرده و از آنها در اپلیکیشن مخصوص پیش‌بینی استفاده می‌کنیم.

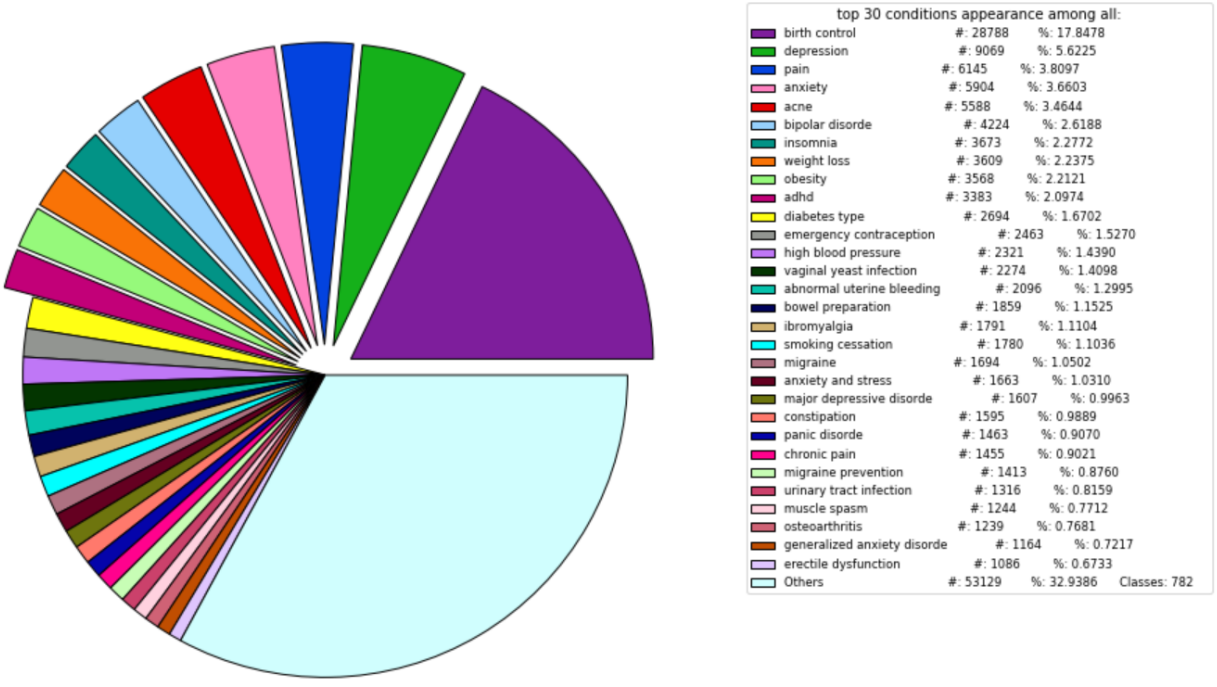
مجموعه داده

مجموعه داده‌ی استفاده شده در این پروژه گرفته شده از داده‌ی ارائه شده در سایت kaggle با نام

UCI ML Drug Review Dataset است که این مجموعه داده در اصل به‌عنوان Drug Review Dataset (Drugs.com) Data Set معرفی شده است. طبق گفته‌ی منبع اصلی این مجموعه داده، این داده‌ها با کرال کردن توضیحات نوشته شده توسط کاربران سایت‌های دارویی گردآوری شده و در سال ۲۰۱۸ میلادی معرفی شده است.به‌صورت کلی این مجموعه داده شامل ۷ ستون شده که در این پروژه تمرکز بر روی ستون‌های زیر است:

- نام دارو (DrugName)
- نام بیماری (condition)
- شرح‌حال (review)

در بخش ابتدایی پروژه که تنها بر روی تشخیص وضعیت (بیماری) بیمار بر اساس پردازش شرح‌حال نوشته شده توسط آن است. با توجه به بالا بودن تنوع بیماری‌های موجود در این مجموعه داده، در بخش اولیه تصمیم بر آن شد تا بر روی تعداد محدودی داده تمرکز شود، بنابراین با بررسی داده‌ها (توضیحات تکمیلی در مورد چگونگی این پردازش در بخش بعدی آورده شده است) توزیع تعداد شرح‌حال‌های موجود برای بیماری‌های متفاوت در داده‌های آموزش در نمودار زیر نمایش داده شده است.



با توجه به این نمودار می‌توان دید که توزیع شرح‌حال‌ها در ۱۰ کلاس اول بیماری(مرتب شده بر اساس بیشترین شرح‌حال موجود برای آن بیماری) از مقدار بالایی برخوردار هستند و پس از فاصله‌ی معناداری در

توزیع آن‌ها را می‌توان شاهد بود به همین منظور تمرکز تنها بر روی ۱۰ بیماری ابتدایی است. در زیر به‌طور دقیق‌تر توزیع داده در این ۱۰ بیماری آورده شده است.

rank	condition	appearance count	percentage among all
1	Birth control	28788	17.8478
2	depresion	9069	5.6225
3	pain	6145	3.8097
4	anxiety	5904	3.6603
5	acne	5588	3.4644
6	bipolar disorder	4224	2.6188
7	insomnia	3673	2.2772
8	weight loss	3609	2.2375
9	obesity	3568	2.2121
10	ADHD	3383	2.0974
∑		73951	45.8477
11	diabetes type	2694	1.6702

که با توجه به جدول بالا می‌توان شاهد فاصله‌ی معنادار بین ۱۰امین بیماری و ۱۱ مین بود.

توابع کلاسها

در این بخش توابع و کلاس‌های معرفی شده در هر فایل از کد پروژه شرح داده می‌شوند.

- فایل consts.py:

  در این فایل متغیرهای ثابتی که در بین دیگر فایل‌ها اشتراک دارند مانند نام بیماری‌های انتخابی و اندیس هر یک از مدل‌های دسته‌بند استفاده شده و ... آورده شده است.

-

- فایل LP_toolkits.py:

این کلاس شامل تابع پردازنده‌ی متن است.

  - تابع : normalizer:

ورودی:

۱. string: یک رشته متن است.

عملکرد:

ابتدا دو فرمت خاص کاراکتر های کوتیشن و دابل کوتیشن را با فرمت درست آن‌ها جاگذاری کرده (زیرا در جدا کردن فشرده‌سازی‌ها دارای اهمیت است). سپس فشرده‌سازی‌های انجام شده را حذف کرده و حالت بازشده‌ی آن‌ها را می‌نویسد مانند مثال زیر:

can't → can not

سپس کاراکترهای اکسنت دار را حذف کرده و به‌فرم کاراکترهای انگلیسی در می‌آورد مانند مثال زیر:

Naïve → naive

و نهایتا حروف اضافه و اعداد را حذف می‌کند و تمام کاراکترهای رشته را به فرم کوچک (lowe case) در آورد و خروجی می‌دهد.

خروجی:

رشته‌ی پردازش شده را به حالت یک رشته خروجی می‌دهد.

- فایل preprocessing.py:

این کلاس شامل تابعی است که پیش‌پردازش‌های لازم برای استفاده از داده‌ی آموزش و آزمایش مورد استفاده را انجام می‌دهد و نمونه‌ای از فراخوانی این تابع در این فایل آورده شده است.

  - تابع get_processed_data:

ورودی:

۱.csv_file_add : آدرس کامل (همراه با نام فایل) داده‌ی csv (یکی از مجموعه داده‌ی آموزش یا آزمایش)

عملکرد:

این تابع ستون‌های مورد نیاز از مجموعه داده را فیلتر کرده و هر یک را با استفاده از تابع normalizer موجود در فایل LP_toolkits.py پردازش کرده و نهایتا آن‌ها در لغت‌نامه‌ای با ساختار زیر ذخیره می‌کند.

{‘review’: <list of normalized reviews strings>,

‘condition’: <list of normalized conditions strings>,

‘Drug’:<list of normalized drugs name strings>}

خروجی:

لغت‌نامه‌ی ساخته شده در این تابع را خروجی می‌دهد.

- فایل data_visualization.ipynb:

ابتدا کتاب‌خانه‌های لازم و توابع موجود در فایل‌های دیگر که این‌جا از آن‌ها استفاده می‌شود را import می‌کنیم. سپس مسیر دو فایل آموزش و آزمایش را (همراه با نام فایل) در دو متغیر با نام‌های مرتبط ذخیره می‌کند. سپس متغیر conditions_selection_num و selected_top را مقدار دهی می‌کند که به‌ترتیب تعداد بیماری‌هایی که پرتکرارترین بوده‌اند و به‌صورت جداگانه در رسم نمودار و چاپ جزئیات در ادامه در نظر گرفته شوند و تعداد بیماری انتخابی برای محدود کردن داده به‌آن تعداد آمده است. (این مقادیر در حالت نهایی به ترتیب برابر ۳۰ و ۱۰ قرار گرفته‌اند)

  - تابع save_processed_data:

ورودی‌ها:

۱. train_data_path: مسیر (همراه با نام فایل) داده‌ی csv آموزش

۲. test_data_path:مسیر (همراه با نام فایل) داده‌ی csv آزمایش

عملکرد:

با صدا زدن تابع get_preprocessed_data از فایل preprocessing.py برای این دو فایل، داده‌ی پردازش شده‌ی آن‌ها را گرفته و در فایلی با نام مناسب برا استفاده در توابع بعدی ذخیره می‌کند.

- تابع conditions_plot:

ورودی:

۱. train_data_dict: لغت‌نامه‌ای که در آن داده‌های آموزش پردازش شده (در فرمت خروجی تابع get_preprocessed_data)

عملکرد:

این تابع با استفاده از لیست مربوط به بیماری‌ها، لغت‌نامه‌ای که برای هر بیماری تعداد تکرار آن به عنوان مقدار (value) بیماری آورده شده است می‌سازد و سپس این لغت‌نامه را بر اساس تعداد تکرار بیماری (که برابر با مقدار در این لغت‌نامه است) به صورت نزولی آن را مرتب می‌کند و با توجه به مقدار conditions_selection_num معرفی شده در ابتدای فایل،این تعداد از بیماری را به‌صورت جداگانه در یک لغت‌نامه ذخیره میی‌کند. سپس با توجه به این لغت‌نامه و لغت‌نامه‌ی اصلی، تعداد ظهور تمامی بیماری‌های دیگر (آن‌هایی که جز این انتخاب شده ها نیستند) و تمامی بیماری‌ها را شمرده و سپس با استفاده از تعداد ظهور conditions_selection_num بیشترین بیماری و تمام بیماری‌های دیگر (با نام others در نمودار) برچسپ های هر دسته از نمودار دایره ای را نام بیماری و مقدار آن را برابر با تعداد ظهور قرار داده و تعداد بیماری انتخابی (selected_top) را از دیگر بیماری ها در نمایش نمودار دایره ای جدا کرده و نهایتا نمودار حاصل را نمایش می دهد.

نتیجه‌ی این نمایش، بردار دایره‌ای آورده شده در قسمت مجموعه داده است.

- تابع conditions_details:

ورودی:

۱. train_data_dict: لغت‌نامه‌ای که در آن داده‌های آموزش پردازش شده (در فرمت خروجی تابع get_preprocessed_data)

عملکرد:

عملکرد این تابع تا زمان یافتن برترین بیماری‌ها و تعداد بیماری‌های جز این بیماری‌های برتر مانند تابع قبل است، اما پس از آن نتیجه را در خروجی چاپ کرده و مجموع تعداد داده بیماری‌های در نظر گرفته شده را همراه با درصد آن‌ها نسب به کل بیماری‌ها نیز نمایش می‌دهد. (قسمتی از این خروجی در جدول آورده شده در قسمت مجموعه داده‌ی این گزارش آورده شده است.)

- فایل Classifiers.py:
  - کلاس PatientDetector:
    - تابع __init__:

ورودی‌ها:

۱. پارامتر self

۲. پارامتر classifier_type: یک int بین ۰ تا ۶ که برای تعیین دسته‌بند مورد نظر است.

عملکرد:

پس دریافت ورودی‌ها، در ابتدا بردارساز و ماژول انتخاب ویژگی را برای شئ تعیین می‌کند. سپس با توجه به عدد ورودی classifier_type، شرط و تابع متناظر با عدد که مخصوص بارگذاری مدل یادگیری ماشین را فراخوانی می‌کند و مدل بازگردانده شده را برای شئ تعیین می‌کند.

- تابع build_logistic_regression_classifier:

ورودی‌ها:

۱. پارامتر self

عملکرد:

یک مدل یادگیری Logistic Regression را initialize کرده و بازمی‌گرداند.

- تابع build_logistic_regression_classifier:

ورودی‌ها:

۱. پارامتر self

عملکرد:

یک مدل یادگیری Logistic Regression را initialize کرده و بازمی‌گرداند.

- تابع build_logistic_regression_classifier:

ورودی‌ها:

۱. پارامتر self

عملکرد:

یک مدل یادگیری Logistic Regression را initialize کرده و بازمی‌گرداند.

- تابع build_svc_classifier:

  ورودی‌ها:

  ۱. پارامتر self

  عملکرد:

  یک مدل یادگیری Support Vector Classifier را initialize کرده و بازمی‌گرداند.

- تابع build_sgd_classifier:

  ورودی‌ها:

  ۱. پارامتر self

  عملکرد:

  یک مدل یادگیری Stochastic Gradient Descent را initialize کرده و بازمی‌گرداند.

- تابع build_decision_tree_classifier:

  ورودی‌ها:

  ۱. پارامتر self

  عملکرد:

  یک مدل یادگیری decision tree را initialize کرده و بازمی‌گرداند.

- تابع build_random_forest_classifeir:

  ورودی‌ها:

  ۱. پارامتر self

  عملکرد:

  یک مدل یادگیری random forest را initialize کرده و بازمی‌گرداند.

- تابع build_adaboost_classifier:

  ورودی‌ها:

  ۱. پارامتر self

  عملکرد:

یک مدل random forest را initialize کرده و بازمی‌گرداند.

- تابع train:

ورودی‌ها:

۱. پارامتر self

۲. پارامتر data_set: لیستی از داده‌های پیش‌پردازش شده است.

عملکرد:

متن شرح‌ها و بیماری‌های را از داده‌ی ورودی گرفته و در لیست‌هایی قرار می‌دهد. سپس به کمک بردارساز، متون را برای یادگیری آماده کرده و در نهایت ان‌ها را به عنوان ورودی به مدل تعیین شده می‌دهیم.

- تابع evaluate:

ورودی‌ها:

۱. پارامتر self

۲. پارامتر data_set: لیستی از داده‌های پیش‌پردازش شده است.

عملکرد:

متن شرح‌ها و بیماری‌های را از داده‌ی ورودی گرفته و در لیست‌هایی قرار می‌دهد. سپس به کمک بردارساز شئ، متون ورودی را به فرمت لازم درآورده و آماده‌ی پیش‌بینی می‌کند. پس از دریافت نتایج پیش‌بینی‌های بیماری‌ها، آنها را با بیماری‌های واقعی مقایسه کرده و به کمک classification report و Confusion matrix ارزیابی می‌کنیم.

- فایل Classifier.py:
  - تابع : main

عملکرد:

داده‌های پیش‌پردازش‌های آموزش و تست را از فایل ذخیره شده خوانده و در لیست‌های مربوط به آنها قرار می‌دهیم. سپس یک شئ از کلاس PatientDetector ساخته و با مدل دلخواه initialize می‌کنیم.سپس به کمک داده‌های فایل آموزش و تابع train از کلاس، مدل را آموزش داده و سپس به کمک داده‌های فایل تست و تابع evaluate، مدل ساخته شده را ارزیابی می‌کنیم.

در نهایت بردارساز مربوط و مدل آموزش داده‌شده را در فایل‌های مربوط ذخیره می‌کنیم.

- فایل app.py: این فایل مرتبط با دمو وبی هست که با استفاده از کتابخانه streamlit پیاده‌سازی کردیم که بعد از اجرا آن می‌تواند به دمویی که بالا می‌آید یک متن به عنوان شرح ورودی بدهید و این دمو نتایج مدل‌های متفاوت را به شما نشان می‌دهد و در این فایل متغیری به نام model_dir_path وجود دارد که برای مشخص کردن مسیر پوشه‌ای است که فایل مدل‌ها در آن ذخیره شده است و نحوه اجرا این دمو هم با دستور زیر است:

steamlit run app.py

ارزیابی و نتایج

فایل ذخیره شده تمامی مدل‌های زیر را می‌توانید از لینک دانلود کنید

- مدل KNN

	precision	recall	f1-score	support
acne	92	82	87	1847
adha	85	68	75	1126
anxiety	54	71	61	1908
bipolar disorder	79	55	65	1380
birth control	90	97	94	9648
depression	70	62	66	3095
insomnia	73	81	77	1231
obesity	62	54	58	1189
pain	83	86	85	2100
weight loss	65	57	61	1248
accuracy			80	24772
macro avg	75	71	73	24772
weighted avg	80	80	80	24772

- مدل SGD

	precision	recall	f1-score	support
acne	89	91	90	1847
adha	87	90	88	1126
anxiety	78	75	76	1908
bipolar disorder	80	77	79	1380
birth control	98	97	97	9648
depression	82	75	79	3095
insomnia	77	91	83	1231
obesity	66	68	67	1189
pain	89	94	92	2100
weight loss	68	69	69	1248
accuracy			87	24772
macro avg	81	83	82	24772
weighted avg	87	87	87	24772

-

- مدل Naive Bayes

	precision	recall	f1-score	support
acne	94	86	90	1135
adha	92	79	85	682
anxiety	76	65	70	1133
bipolar disorder	88	68	77	824
birth control	95	98	97	5766
depression	67	85	75	1790
insomnia	86	85	86	694
obesity	68	64	66	741
pain	93	90	92	1275
weight loss	70	71	70	751
accuracy			86	14791
macro avg	83	79	81	14791
weighted avg	86	86	86	14791

-

- مدل Logistic Regression

	precision	recall	f1-score	support
acne	95	89	92	1847
adha	94	85	90	1126
anxiety	80	75	77	1908
bipolar disorder	86	74	80	1380
birth control	97	98	98	9648
depression	77	85	81	3095
insomnia	87	87	87	1231
obesity	71	69	70	1189
pain	91	94	93	2100
weight loss	73	69	71	1248
accuracy			89	24772
macro avg	85	83	84	24772
weighted avg	89	89	88	24772

-

- مدل SVM

	precision	recall	f1-score	support
acne	93	96	94	1847
adha	97	93	95	1126
anxiety	85	86	85	1908
bipolar disorder	91	87	89	1380
birth control	99	97	98	9648
depression	85	89	87	3095
insomnia	88	91	89	1231
obesity	85	83	84	1189
pain	94	95	95	2100
weight loss	85	85	85	1248
accuracy			93	24772
macro avg	90	90	90	24772
weighted avg	93	93	93	24772

-

- مدل Decision Tree

	precision	recall	f1-score	support
acne	93	92	93	1847
adha	90	90	90	1126
anxiety	80	78	79	1908
bipolar disorder	79	81	80	1380
birth control	96	96	96	9648
depression	81	83	82	3095
insomnia	82	80	81	1231
obesity	78	80	79	1189
pain	90	90	90	2100
weight loss	79	76	78	1248
accuracy			89	24772
macro avg	85	85	85	24772
weighted avg	89	89	89	24772

-

- مدل Random Forest

	precision	recall	f1-score	support
acne	98	94	96	1847
adha	97	92	94	1126
anxiety	89	83	86	1908
bipolar disorder	95	83	88	1380
birth control	95	99	97	9648
depression	86	91	88	3095
insomnia	90	89	89	1231
obesity	87	81	84	1189
pain	94	95	95	2100
weight loss	86	83	85	1248
accuracy			93	24772
macro avg	92	89	90	24772
weighted avg	93	93	93	24772

-

- مدل AdaBoost

	precision	recall	f1-score	support
acne	62	92	74	1847
adha	86	77	81	1126
anxiety	74	56	64	1908
bipolar disorder	81	57	67	1380
birth control	96	85	90	9648
depression	54	75	63	3095
insomnia	71	79	74	1231
obesity	53	56	54	1189
pain	88	83	85	2100
weight loss	61	56	58	1248
accuracy			77	24772
macro avg	73	72	71	24772
weighted avg	79	77	77	24772

تقسیم‌بندی کارها

	شیده	امیرعلی	مهراد
تمیز و آماده‌سازی مجموعه داده	95%	5%	
پیاده‌سازی مدل‌ها		50%	50%
اپلیکیشن	40%	40%	20%
ارائه و تهیه گزارش	33%	33%	33%