پردیس علوم
دانشکده ریاضی، آمار و علوم کامپیوتر

# بررسی رابطه بین متن نوشته شده توسط یک فرد و خودکشی کردن او با استفاده از متد های کاوش متن و یادگیری ماشین

نگارنده

**نرگس بابااحمدی**

استاد راهنما: دکتر باقر باباعلی

پایان‌نامه برای دریافت درجه کارشناسی
در رشته علوم کامپیوتر

تیر ۱۴۰۲

# چکیده

خودکشی یکی از چالش‌های اصلی سلامت عمومی است و شناسایی زودهنگام افراد در معرض خطر خودکشی می‌تواند گامی مهم در پیشگیری از آن باشد. با این حال، شناسایی افراد در معرض خطر خودکشی یک وظیفه چالش برانگیز است زیرا رفتار این افراد معمولاً پدیده پیچیده و چند وجهی است. در سال‌های اخیر، تکنیک‌های داده کاوی متن [1] و یادگیری ماشین[2] برای تجزیه و تحلیل داده‌های متنی و شناسایی الگوهای نشان دهنده رفتار خودکشی استفاده شده است.
هدف این پایان‌نامه، توسعه یک مدل با دقت بالا برای شناسایی متونی که نشانگر رفتار منجر به خودکشی هستند با استفاده از تکنیک‌های داده کاوی متن و یادگیری ماشین است. به منظور دستیابی به این هدف، از داده‌های متنی موجود در رسانه اجتماعی ردیت [3] استفاده شده است، زیرا این رسانه یک منبع غنی از داده‌های متنی است که می‌تواند برای تحلیل زبان افرادی که ممکن است در معرض خطر خودکشی باشند، استفاده شود. ترکیبی از تکنیک‌های یادگیری ماشین و یادگیری عمیق شامل ماشین بردار پشتیبان، جنگل تصادفی و شبکه‌های عصبی برای توسعه مدل استفاده شده است.
این پایان‌نامه می‌تواند به عنوان پایه‌ای برای تحقیقات بیشتر در این حوزه و ارائه راهکارهای بهبود سلامت عمومی افراد مورد استفاده قرار گیرد.

[1] Text Mining
[2] Machine Learning
[3] Reddit

# سپاسگزاری

از استاد راهنمای گرانقدر جناب آقای دکتر باباعلی که با راهنمایی های خود مرا در انجام این پروژه یاری دادند کمال تشکر را دارم.

# پیشگفتار

موضوع خودکشی یکی از مهمترین موضوعاتی است که در حوزه سلامت عمومی بررسی می‌شود. خودکشی یکی از عوامل مهم در کاهش کیفیت زندگی افراد و حتی ایجاد مشکلات جدی برای جامعه است. با توجه به اینکه خودکشی به صورت غیرمستقیم و بدون هشدار قبلی اتفاق می‌افتد، شناسایی زودهنگام آن بسیار حائز اهمیت است.

امروزه، با پیشرفت فناوری و کاهش هزینه‌های داده کاوی و یادگیری ماشین، استفاده از این روش‌ها به عنوان یک راهکار جدید در شناسایی افراد در معرض خطر خودکشی مطرح شده است. در این راستا، تحلیل داده‌های متنی می‌تواند به عنوان یکی از روش‌های مؤثر در شناسایی افراد در معرض خطر خودکشی مورد استفاده قرار گیرد.

در این پایان نامه، با بررسی ادبیات و تحلیل داده‌های موجود در پلتفرم‌های آنلاین، مدلی با دقت بالا برای شناسایی افراد در معرض خطر خودکشی پیشنهاد شده است. برای این منظور، داده‌های متنی مختلف از جمله متون نوشته شده توسط افراد در شبکه‌های اجتماعی، ایمیل‌های ارسال شده و پیام‌های متنی ارسال شده به دیگران، جمع‌آوری و تحلیل شده‌اند. سپس با استفاده از متدهای یادگیری ماشین و همچنین شبکه‌های عصبی [۴] ، الگوهای مشترک در این متون شناسایی شده و مدلی برای شناسایی افراد در معرض خطر توسعه داده شده است.

[4] Neural Networks

# فهرست مطالب

# فصل ۱

# مفاهیم مقدماتی

## ۱.۱ مقدمه ای بر یادگیری ماشین

یادگیری ماشین به عنوان یکی از حوزه‌های مهم هوش مصنوعی [۱] ، امکانات بسیاری را برای ما فراهم می‌کند. این روزها، بسیاری از کارها و اتفاقات در اطراف ما، توسط ماشین‌ها و بدون دخالت انسان‌ها انجام م ی‌شودو بعضا بازدهی بیشتری و عملکرد دقیق‌تری نسبت به انسان‌ها دارند [۱] .
آیا این به معنای این است که ماشین‌ها فکر می کنند؟ آلن تورینگ، با مطرح کردن این سوال در یکی از مقالات خود در سال ۱۹۵۰، نقطه شروعی برای پژوهش‌ها و فعالیت‌های در زمینه هوش مصنوعی قرار داد.
در دهه‌ی ۱۹۶۰، پیدایش شبکه‌های عصبی ، باعث شد که یادگیری ماشین به یک سطح جدید از پیچیدگی و کاربرد برسد. در این دهه، الگوریتم‌های جدیدی نیز مانند الگوریتم‌های بیزی [۲] و الگوریتم‌های خوشه‌بندی [۳] ، ارائه شدند.
در دهه‌ی ۱۹۹۰، با پیدایش اینترنت و تولید داده‌های بزرگ، اهمیت یادگیری ماشین به طور چشمگیری افزایش یافت. در این دهه، الگوریتم‌های جدیدی نیز مانند شبکه‌های عصبی عمیق [۴] ، الگوریتم‌های ماشین بردار پشتیبان [۵] و الگوریتم‌های شبکه‌های مولد [۶] ، ارائه شدند[۲]. در دهه‌ی اخیر، با پیشرفت تکنولوژی‌های پردازشی و تولید داده‌های بیشتر، یادگیری ماشین به یکی از حیاتی‌ترین حوزه‌های علمی و کاربردی تبدیل شده است. به طور کلی، این حوزه به دلیل کاربردهای فراوانی که در حوزه‌های مختلف دارد، همچنان در حال توسعه و پیشرفت است.

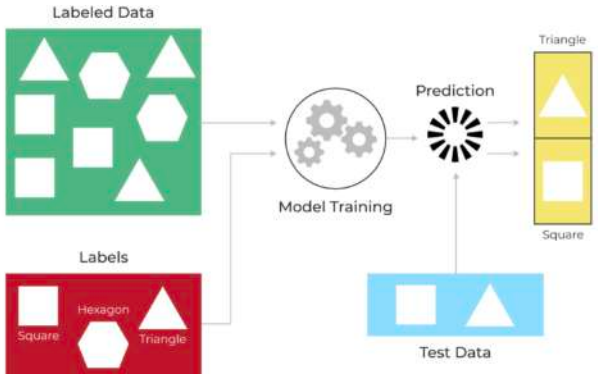
[1] Artificial Intelligence
[2] Bayesian Algorithms
[3] Clustering Algorithms
[4] Deep Neural Networks
[5] Support Vector Machine
[6] Generative Adversarial Networks

این پروژه با استفاده از متدهای یادگیری ماشین و یادگیری عمیق انجام شده است. در ادامه به تعدادی از این مفاهیم می‌پردازیم.

## ۱.۱.۱ الگوریتم های یادگیری ماشین

### یادگیری نظارت شده

الگوریتم یادگیری نظارت شده [۷] یکی از رایج‌ترین الگوریتم‌های یادگیری ماشین است. در این الگوریتم، مدل یادگیری ماشین از داده‌های آموزشی با برچسب استفاده می‌کند تا بتواند روی داده‌های جدیدی که برچسب ندارند، پیش‌بینی کند[۳].
در این الگوریتم، داده‌ها به دو بخش تقسیم می‌شوند: داده‌های آموزشی و داده‌های تست. داده‌های آموزشی شامل ورودی‌هایی است که برچسب یا پاسخ صحیح دارند، به عنوان مثال، تصاویری از حیوانات با برچسب‌های "گربه" و "سگ". در این الگوریتم، مدل یادگیری ماشین از داده‌های آموزشی استفاده می‌کند تا یاد بگیرد که هر نوع تصویر، چه برچسبی دارد[۴]. سپس با استفاده از این دانش، می‌تواند برای تصاویر جدیدی که برچسب ندارند، پیش‌بینی کند که برچسب آن ها چیست[۵].




شکل ۱.۱: یادگیری نظارت شده

برای آموزش مدل یادگیری نظارت شده، ابتدا باید یک تابع هدف [۸] انتخاب کرد. این تابع باید درستی پیش‌بینی مدل را به عنوان یک عدد تخمینی برای هر داده آموزشی محاسبه کند. سپس مدل با استفاده از روش‌های بهینه‌سازی، مانند کاهش گرادیان [۹]، سعی می‌کند تا این تابع هدف را بیشینه

[7] Supervised Learning
[8] Objective Function
[9] Gradient Descent

کند. با این کار، دقت پیش‌بینی مدل برای داده‌های آموزشی افزایش می‌یابد.
بعد از آموزش مدل با داده‌های آموزشی، می‌توان از آن برای پیش‌بینی برچسب داده‌های تست استفاده کرد. با این کار، می‌توان دقت پیش‌بینی مدل را برای داده‌های جدید ارزیابی کرد. در صورتی که دقت پیش‌بینی مدل برای داده‌های تست نیز به اندازه کافی بالا باشد، می‌توان آن را برای پیش‌بینی برچسب داده‌های جدید استفاده کرد.
چند نمونه از الگوریتم‌های یادگیری نظارت شده عبارتند از [۶] :

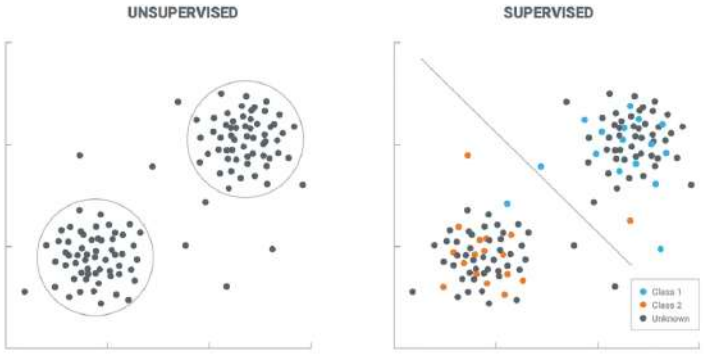
- الگوریتم درخت تصمیم [۱۰]
- الگوریتم نزدیک‌ترین همسایه [۱۱]
- الگوریتم ماشین بردار پشتیبان
- الگوریتم رگرسیون خطی [۱۲]
- الگوریتم رگرسیون لجستیک [۱۳]

### یادگیری نظارت نشده

الگوریتم یادگیری نظارت نشده [۱۴] یک الگوریتم یادگیری ماشین است که در آن داده‌هایی بدون برچسب به مدل یادگیری داده می‌شوند و مدل باید الگوهایی که در داده‌ها وجود دارد را شناسایی کند. در این الگوریتم، مدل به صورت خودکار الگوهایی را کشف می‌کند که در داده‌ها وجود دارد و این الگوها را برای تکرار در آینده استفاده می‌کند[۷].
در الگوریتم یادگیری نظارت نشده، داده‌ها به دو بخش تقسیم می‌شوند: داده‌های آموزشی و داده‌های تست. در بخش آموزشی، مدل با استفاده از داده‌های بدون برچسب، سعی می‌کند الگوهایی را که در داده‌ها وجود دارد، شناسایی کند. به عنوان مثال، در صورت داشتن تصاویری از طبیعت بدون برچسب، مدل باید بتواند الگوهایی مانند درختان، آسمان، آفتاب، آسیاب‌های بادی و غیره را شناسایی کند. برای این منظور، از الگوریتم‌های یادگیری نظارت نشده مانند مدل‌های خوشه‌بندی استفاده می‌شود.
یکی از روش‌های مهم یادگیری نظارت نشده در داده‌های متنی، استفاده از مدل‌های مبتنی بر ورودی است. در این روش، برای هر کلمه در داده‌های متنی، یک بردار ویژگی مشخص می‌شود که شامل ویژگی‌هایی مانند تعداد تکرار کلمه در متن، تعداد کلمات هم‌معنی، تعداد کلمات مشابه در متن و

[10] Decision Tree
[11] k-Nearest Neighbors
[12] Linear Regression
[13] Logistic Regression
[14] Unsupervised Learning

غیره است. با استفاده از این بردار ویژگی، می‌توان با الگوریتم‌های خوشه‌بندی ، مدل‌های یادگیری ماشین را آموزش داد و به شناسایی الگوها و الگوهایی که در داده‌ها وجود دارد، پرداخت.



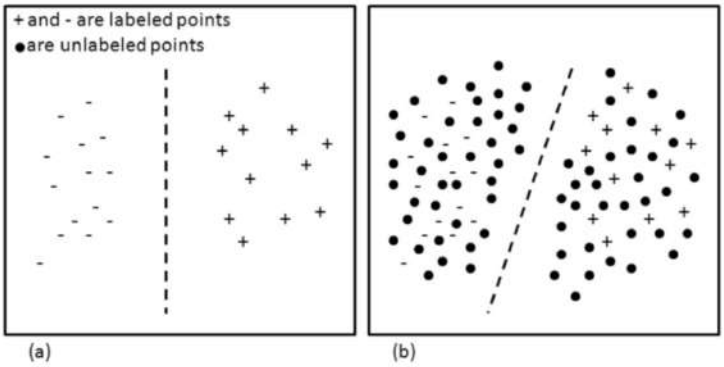

شکل ۲.۱: مقایسه یادگیری نظارت شده و نظارت نشده

## یادگیری نیمه نظارتی

یادگیری نیمه نظارتی [۱۵] یک شیوه یادگیری است که در آن، برخلاف یادگیری نظارتی که دارای برچسب برای داده‌ها است، برخی از داده‌ها بدون برچسب و برخی دیگر دارای برچسب هستند. هدف در یادگیری نیمه نظارتی، تخمین برچسب داده‌های بدون برچسب با استفاده از داده‌های دارای برچسب است.
یادگیری نیمه نظارتی در بسیاری از موارد، به دلیل اینکه بدون نیاز به برچسب داده‌های بیشتر، قابل اجرا است، بسیار مفید است. به عنوان مثال، در بسیاری از مسائل دسته‌بندی، بعضی از داده‌ها دارای برچسب هستند و بعضی دیگر ندارند. با استفاده از داده‌های دارای برچسب، می‌توان یک مدل یادگیری نظارتی ایجاد کرد و با استفاده از آن، برای داده‌های بدون برچسب، برچسب تخمین زد.
برای یادگیری نیمه نظارتی، روش‌های استفاده از داده‌های دارای برچسب و بدون برچسب وجود دارد. یکی از روش‌های معمول، استفاده از شبکه‌های عصبی با نوع خاصی از لایه‌ها است که به نام لایه‌های نیمه نظارتی [۱۶] معروفند. این لایه‌ها به گونه‌ای طراحی شده‌اند که بتوانند از داده‌های دارای برچسب و بدون برچسب به طور همزمان استفاده کنند و مدلی را ایجاد کنند که بتواند برای داده‌های بدون برچسب، برچسب تخمینی تولید کند.
یکی دیگر از روش‌های معمول در یادگیری نیمه نظارتی، استفاده از الگوریتم‌های خوشه‌بندی است.

[15] Semi-Supervised Learning
[16] Semi-Supervised Layers

در این روش، ابتدا داده‌های دارای برچسب به گروه‌های خاصی تقسیم می‌شوند و سپس داده‌های بدون برچسب با استفاده از خوشه‌بندی، در گروه‌هایی با شباهت بالا قرار می‌گیرند. سپس برای هر گروه، یک مدل یادگیری نظارتی ایجاد می‌شود و با استفاده از این مدل‌ها، برای داده‌های بدون برچسب، برچسب تخمینی تولید می‌شود.
در کل، یادگیری نیمه نظارتی یک شیوه یادگیری مفید برای مسائلی است که دارای داده‌های بدون برچسب هستند و یا به دلیل هزینه‌ی بالای برچسب‌گذاری داده‌ها، امکان استفاده از داده‌های بیشتر با برچسب نیست. با استفاده از این شیوه یادگیری، می‌توان به دقت و کارایی بهتری رسید و در بسیاری از موارد، نیاز به داده‌های بیشتر با برچسب را کاهش داد.




شکل ۳.۱: مقایسه یادگیری نظارت شده (a) و نیمه نظارتی (b)

## یادگیري تقویتی

یادگیری تقویتی [۱۷] در واقع یکی از شاخه‌های مهم یادگیری ماشین است که به شیوه تعاملی و با دنبال کردن یک رویه عملکرد، عامل یادگیری ماشین با محیط تعامل دارد و با هدف حصول بیشینه پاداش از محیط، به یادگیری می‌پردازد. در واقع، یادگیری تقویتی به عنوان یک فرآیند تعاملی است که در آن، عامل باید با اعمال یک عمل به محیط، بیشینه پاداش را به دست آورد.
در یادگیری تقویتی، عامل با دیدن وضعیت فعلی محیط، با توجه به سیاستی که دارد، یک عمل را انتخاب می‌کند و برای اعمال آن به محیط، پاداشی از محیط دریافت می‌کند. با گذشت زمان و انجام اعمال مختلف، عامل می‌تواند با تجربه، یاد بگیرد که در وضعیت‌های مختلف، کدام عمل بهترین پاداش را با خود به همراه دارد.
در یادگیری تقویتی، محیط می‌تواند به صورت یک بازی، یک ربات صنعتی، یا حتی یک محیط

[17] Reinforcement Learning

مجازی باشد. وضعیت، شرایط فعلی محیط را توصیف می‌کند. برای مثال، در یک بازی، وضعیت شامل موقعیت بازیکن، تعداد جان بازیکن، امتیاز بازیکن و غیره است. عمل، هرگونه اقدام یا تصمیمی است که عامل در محیط انجام می‌دهد. برای مثال، در یک بازی، عمل می‌تواند شامل حرکت کردن بازیکن، حمله به حریف و غیره باشد.
در یادگیری تقویتی، سیاست، مجموعه‌ای از قوانین و رویه‌هایی است که عامل با توجه به وضعیت جاری، یک عمل را انتخاب می‌کند. پاداش، امتیازی است که عامل به دست می‌آورد و اندازه‌گیری می‌کند که عملی که انجام داده است، مفید واقع شده است یا خیر. تابع ارزش، یک تخمین از میزان پاداشی است که می‌توان با انجام یک عمل در یک وضعیت خاص، به دست آورد. این تابع برای هر وضعیت و برای هر عمل، یک ارزش تخمینی را محاسبه می‌کند که برای یادگیری تقویتی بسیار مهم است[۸].

### ۲.۱.۱ الگوریتم‌های طبقه‌بندی

الگوریتم‌های طبقه‌بندی [۱۸] الگوریتم‌هایی هستند که برای مسئله دسته‌بندی داده‌ها به کلاس‌های مختلف استفاده می‌شوند. در ادامه، به برخی از این الگوریتم‌ها اشاره می‌شود.

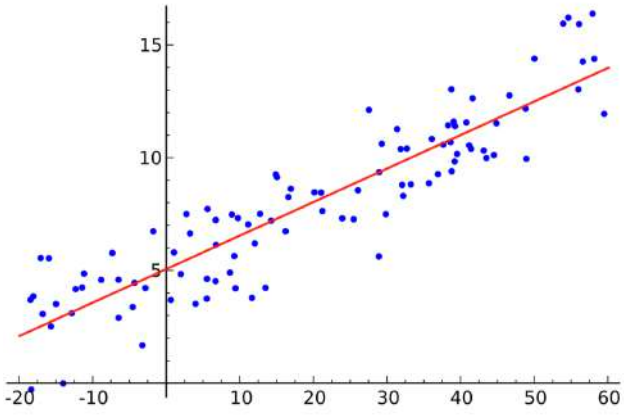
#### رگرسیون خطی ساده

رگرسیون [۱۹] روشی برای مدل‌سازی یک مقدار هدف بر اساس پیش‌بینی‌کننده های مستقل است. این روش بیشتر برای پیش‌بینی و یافتن رابطه علت و معلولی بین متغیرها استفاده می‌شود. تکنیک‌های رگرسیون عمدتاً بر اساس تعداد متغیرهای مستقل و نوع رابطه بین متغیرهای مستقل و وابسته متفاوت است.
رگرسیون خطی ساده نوعی تحلیل رگرسیونی است که در آن تعداد متغیرهای مستقل یک است و بین متغیر مستقل (x) و متغیر وابسته (y) رابطه خطی وجود دارد. خط قرمز در نمودار ۴.۱به عنوان بهترین برازش خط مستقیم نامیده می‌شود. بر اساس نقاط داده شده، سعی می‌کنیم خطی را ترسیم کنیم که نقاط را به بهترین شکل برازش کند. خط را می توان بر اساس معادله خطی نشان داده شده در زیر مدل کرد[۹].

$$y = \alpha + \beta x \qquad (۱.۱)$$

انگیزه الگوریتم رگرسیون خطی، یافتن بهترین مقادیر برای $\alpha$ و $\beta$ است. برای درک بهتر رگرسیون خطی، به دو مفهوم مهم، نگاهی می‌اندازیم.

[18] Classification Algorithms
[19] Regression




شکل ۴.۱: بهترین برازش خطی

- تابع هزینه به ما کمک می کند تا بهترین مقادیر ممکن را برای $\alpha$ و $\beta$ پیدا کنیم که بهترین خط را برای نقاط داده برازش می‌کند. از آنجایی که ما بهترین مقادیر را برای $\alpha$ و $\beta$ می‌خواهیم، این مساله جستجو را به یک مساله کمینه‌سازی تبدیل می‌کنیم که در آن می‌خواهیم خطای بین مقدار پیش‌بینی‌شده و مقدار واقعی را به حداقل برسانیم.

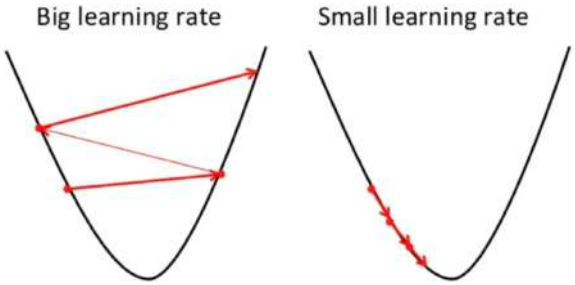
$$minimize \frac{۱}{n} \sum_{i=۱}^{n} (pred_i - y_i)^۲ \qquad (۲.۱)$$

$$J = \frac{۱}{n} \sum_{i=۱}^{n} (pred_i - y_i)^۲ \qquad (۳.۱)$$

تابع بالا را برای کمینه سازی انتخاب می کنیم. تفاوت بین مقادیر پیش‌بینی‌شده و واقعی، خطا را اندازه‌گیری می‌کند. خطا را مربع می کنیم و روی تمام نقاط داده جمع می کنیم و آن مقدار را بر تعداد کل نقاط داده تقسیم می‌کنیم. این میانگین، مربعات خطا را در تمام نقاط داده ارائه می‌دهد. بنابراین، این تابع هزینه به عنوان تابع میانگین مربعات خطا [۲۰] نیز شناخته می شود. حال با استفاده از این تابع می‌خواهیم مقادیر $\alpha$ و $\beta$ را طوری تغییر دهیم که مقدار مربعات خطا در مینیمم قرار گیرد[۱۰].

[20] MSE

- مفهوم مهم دیگر برای درک رگرسیون خطی، کاهش گرادیان است. کاهش گرادیان روشی برای به روز رسانی $\alpha$ و $\beta$ برای کاهش تابع هزینه است. ایده این است که ما با مقادیری برای $\alpha$ و $\beta$ شروع می کنیم و سپس این مقادیر را به طور مکرر تغییر می‌دهیم تا هزینه را کاهش دهیم. کاهش گرادیان به ما در چگونگی تغییر مقادیر کمک می‌کند[۱۱].




شکل ۵.۱: نرخ یادگیری

برای ترسیم قیاس، گودالی را به شکل U تصور می‌کنیم که در بالاترین نقطه گودال ایستاده‌ایم و هدف ما رسیدن به انتهای گودال است. مساله‌ای وجود دارد که ما فقط می‌توانیم تعداد محدودی قدم برداریم تا به پایین گودال برسیم. اگر تصمیم بگیریم هر بار یک قدم برداریم، در نهایت به انتهای گودال خواهیم رسید، اما این کار زمان زیادی می‌برد. اگر تصمیم بگیریم هر بار قدم های بلندتری برداریم، زودتر به آن می رسیم، اما این احتمال وجود دارد که از پایین گودال عبور کنیم و دقیقاً در انتهای گودال توقف نکنیم. در الگوریتم کاهش گرادیان، تعداد قدم‌هایی که برمی‌داریم، نرخ یادگیری است. این نرخ تصمیم می‌گیرد که الگوریتم با چه سرعتی به حداقل همگرا شود.
برای به روز رسانی مقادیر $\alpha$ و $\beta$، گرادیان تابع هزینه را محسابه می‌کنیم. برای یافتن این گرادیان‌ها، مشتقات جزئی را با توجه به $\alpha$ و $\beta$ محاسبه می‌کنیم[۱۲].

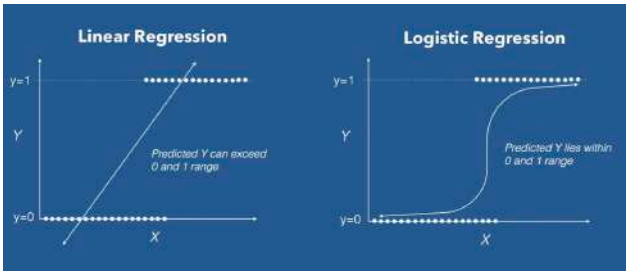
$$\alpha := \alpha - l\frac{۲}{n}\sum_{i=۱}^{n}(pred_i - y_i) \qquad (۴.۱)$$

$$\beta := \beta - l\frac{۲}{n}\sum_{i=۱}^{n}(pred_i - y_i).x_i \qquad (۵.۱)$$

مشتقات جزئی گرادیان هستند و برای به روز رسانی مقادیر $\alpha$ و $\beta$ استفاده می‌شوند. $l$ نرخ یادگیری است که یک ابرپارامتر است که باید آن را مشخص کنیم. نرخ یادگیری کمتر می‌تواند ما را به حداقل نزدیک‌تر کند، اما زمان بیشتری برای رسیدن به حداقل طول می‌کشد، نرخ یادگیری بزرگ‌تر زودتر همگرا می‌شود، اما این احتمال وجود دارد که از حداقل‌ها عبور کنیم.

### رگرسیون لجستیک

رگرسیون لجستیک یک الگوریتم یادگیری ماشین است که برای مسائل طبقه‌بندی استفاده می‌شود، یک الگوریتم تحلیل پیش‌بینی و مبتنی بر مفهوم احتمال است.



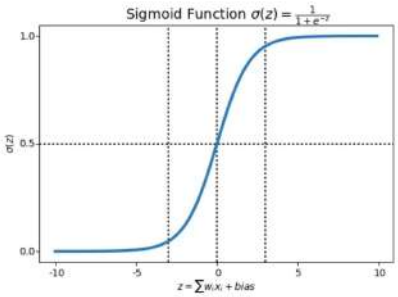

شکل ۶.۱: رگرسیون خطی و لجستیک

می‌توانیم رگرسیون لجستیک را یک مدل رگرسیون خطی بنامیم، اما رگرسیون لجستیک از یک تابع هزینه پیچیده‌تر استفاده می‌کند، این تابع هزینه را می‌توان به‌عنوان «تابع سیگموید» [21]یا به‌جای تابع خطی به‌عنوان «تابع لجستیک» تعریف کرد.
فرضیه رگرسیون لجستیک این است که تابع هزینه را بین ۰ و ۱ محدود کند. بنابراین توابع خطی نمی‌توانند مورد استفاده قرار گیرند، زیرا این توابع می‌توانند مقداری بیشتر از ۱ یا کمتر از ۰ داشته باشد که طبق فرضیه رگرسیون لجستیک امکان پذیر نیست[۱۳].

$$۰ \leq h_\theta(x) \leq ۱ \qquad (۶.۱)$$

[21]Sigmoid function

برای نگاشت مقادیر پیش بینی شده به احتمالات، از تابع سیگموید استفاده می‌کنیم. تابع هر مقدار واقعی را به مقدار دیگری بین ۰ و ۱ نگاشت می‌کند. در یادگیری ماشین، از تابع سیگموید برای ترسیم پیش بینی‌ها به احتمالات استفاده می‌کنیم.




شکل ۷.۱: تابع سیگموید

همانطور که در مورد تابع هزینه ($J_\theta$) در رگرسیون خطی بیان شد، این تابع نشان دهنده هدف بهینه‌سازی است، یعنی یک تابع هزینه ایجاد می‌کنیم و آن را به حداقل می‌رسانیم تا بتوانیم یک مدل دقیق با حداقل خطا ایجاد کنیم.

$$J(\theta) = \frac{۱}{۲}\sum_{i=۱}^{m}(h_\theta(x^i) - y^i)^۲ \qquad (۷.۱)$$

اگر بخواهیم از تابع هزینه رگرسیون خطی در رگرسیون لجستیک استفاده کنیم، فایده‌ای نخواهد داشت زیرا در نهایت یک تابع غیرمحدب با کمینه‌های موضعی [۲۲] متعدد است، که در آن به حداقل رساندن مقدار تابع هزینه و پیدا کردن کمینه کلی [۲۳]، بسیار دشوار است.
تابع هزینه در رگرسیون لجستیک به این صورت تعریف می‌شود:

$$Cost(h_\theta(x), y) = \begin{cases} -\log(h_\theta(x)), & \text{اگر } y = ۱ \\ -\log(۱ - h_\theta(x)), & \text{اگر } y = ۰ \end{cases}, \qquad (۸.۱)$$

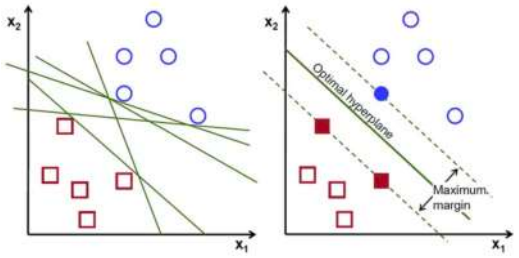
تابع فوق را می‌توان در فرم بسته نیز نوشت:

$$J(\theta) = \frac{-۱}{m}\Sigma[y^i \log(h_\theta(x(i))) + (۱ - y^i)\log(۱ - h_\theta(x(i)))] \qquad (۹.۱)$$

[22] Local Minimums
[23] Global Minimum

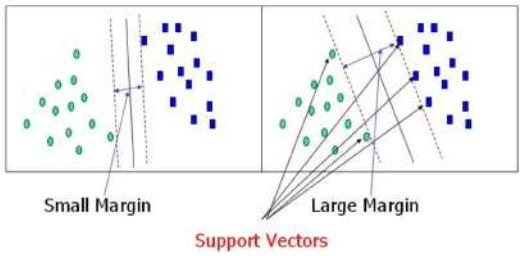
### ماشین بردار پشتیبان

هدف از الگوریتم ماشین بردار پشتیبان، یافتن یک ابر صفحه [۲۴]در یک فضای N بعدی است که به طور مشخص نقاط داده را طبقه‌بندی می‌کند.
برای جدا کردن دو دسته از نقاط داده، ابرصفحه‌های زیادی وجود دارد که می‌توان انتخاب کرد. هدف ما این است که صفحه‌ای را پیدا کنیم که حداکثر حاشیه، یعنی حداکثر فاصله بین نقاط داده هر دو کلاس را داشته باشد. حداکثر کردن فاصله حاشیه، [۲۵]تقویتی بوجود می‌آورد که بتوان نقاط داده آینده را با اطمینان بیشتری طبقه‌بندی کرد[۱۴].




شکل ۸.۱: ابرصفحه‌های ممکن [۱۵]

ابرصفحه‌ها، مرزهای تصمیم‌گیری هستند که به طبقه‌بندی نقاط داده کمک می‌کنند. نقاط داده‌ای که در هر دو طرف ابرصفحه قرار می‌گیرند را می توان به کلاس‌های مختلف نسبت داد. همچنین، ابعاد ابرصفحه به تعداد ویژگی‌ها بستگی دارد. اگر تعداد ویژگی‌های ورودی ۲ باشد، آنگاه ابرصفحه فقط یک خط است. اگر تعداد ویژگی‌های ورودی ۳ باشد، ابرصفحه به یک صفحه دو بعدی تبدیل می‌شود. تصور زمانی که تعداد ویژگی‌ها از ۳ بیشتر شود دشوار می‌شود.




شکل ۹.۱: بردار پشتیبان

[24]Hyperplane
[25]Margin Distance

بردارهای پشتیبان نقاط داده‌ای هستند که به ابرصفحه نزدیکتر می‌باشند و بر موقعیت و جهت ابرصفحه تأثیر می‌گذارند. با استفاده از این بردارهای پشتیبان، حاشیه طبقه بندی‌کننده را به حداکثر می‌رسانیم. حذف بردارهای پشتیبان موقعیت ابرصفحه را تغییر می‌دهد. این‌ها نکاتی هستند که به ما در ساخت ماشین بردار پشتیبان کمک می‌کنند.
در رگرسیون لجستیک، خروجی تابع خطی را می‌گیریم و با استفاده از تابع سیگموید مقدار را در بازه ۰ تا ۱ قرار می‌دهیم. اگر مقدار به دست آمده بزرگتر از مقدار آستانه $\frac{۱}{۲}$ باشد، به آن برچسب ۱ و در غیر این صورت برچسب ۰ به آن اختصاص می‌دهیم. در ماشین بردار پشتیبان، خروجی تابع خطی را می‌گیریم و اگر آن خروجی بزرگتر از ۱ باشد، شناسایی می‌کنیم. آن را با یک کلاس و اگر خروجی منفی ۱ باشد، با کلاس دیگری شناسایی می کنیم.
حال با توجه به مطالب ذکر شده در بالا، تابع ضرر و گرادیان ماشین بردار پشتیبان به صورت زیر بدست می‌آیند[۱۶]:

$$\min_{w} \lambda||w||^{۲} + \Sigma_{i=۱}^{n}(۱ - y_i < x_i, w >)_+ \qquad (۱۰.۱)$$

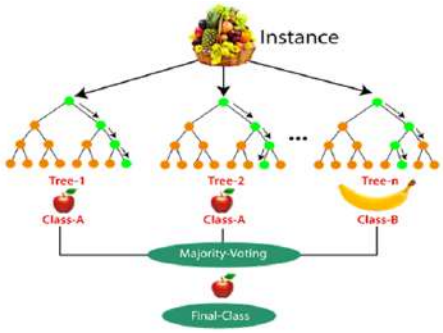
$$w := w + l.(y_i.x_i - ۲\lambda w) \qquad (۱۱.۱)$$

### الگوریتم جنگل تصادفی

الگوریتم جنگل تصادفی [۲۶] یکی از الگوریتم‌های یادگیری ماشین است که به تکنیک یادگیری نظارت شده تعلق دارد. این الگوریتم برای مسائل طبقه‌بندی و رگرسیون در یادگیری ماشین مورد استفاده قرار می‌گیرد. اصولاً این الگوریتم بر مبنای مفهوم یادگیری گروهی عمل می‌کند، به این معنا که چندین طبقه‌بندی‌کننده را با هم ترکیب می‌کند تا یک مسئله پیچیده را حل کرده و عملکرد مدل را بهبود بخشد.
در الگوریتم جنگل تصادفی، چندین درخت تصمیم در زیرمجموعه‌های مختلف از مجموعه داده قرار می‌گیرند و به صورت میانگین، خروجی طبقه‌بندی را بهبود می‌بخشند. به عبارت دیگر، این الگوریتم به جای اعتماد به یک درخت تصمیم، از پیش‌بینی‌های هر درخت و بر اساس اکثریت آرا، پیش‌بینی نهایی را انجام می‌دهد. تعداد بیشتر درختان در جنگل، دقت پیش‌بینی را افزایش می‌دهد و از بروز مشکل بیش‌برازش[۲۷] جلوگیری می‌کند [۱۷] .
این الگوریتم در دو فاز قابل بررسی است. فاز اول ایجاد جنگل تصادفی با ترکیب N درخت تصمیم و فاز دوم پیش‌بینی برای هر درخت ایجاد شده در فاز اول است. کارکرد این الگوریتم را می‌توان در پنج مرحله توصیح داد:

[26] Random Forest Algorithm
[27] Overfitting

مرحله ۱: نقاط تصادفی K داده را از مجموعه آموزش انتخاب می‌کنیم.
مرحله ۲: درختان تصمیم مرتبط با نقاط داده انتخاب شده (زیرمجموعه‌ها) را می‌سازیم.
مرحله ۳: تعداد N را برای درختان تصمیمی که می‌خواهیم بسازیم انتخاب می‌کنیم.
مرحله ۴: مرحله ۱ و ۲ را تکرار می‌کنیم.
مرحله ۵: برای نقاط داده جدید، پیش‌بینی‌های هر درخت تصمیم‌گیری را پیدا می‌کنیم و نقاط داده جدید را به دسته‌ای که آرای اکثریت را کسب می‌کند، اختصاص می‌دهیم.




شکل ۱۰.۱: کارکرد الگوریتم جنگل تصادفی

### دسته‌بند بیز

دسته‌بند بیز [۲۸] مدل یادگیری ماشین بر پایه احتمال است که برای طبقه‌بندی استفاده می‌شود. اصل طبقه‌بندی‌کننده بر اساس قضیه بیز است.

**قضیه بیز:** با استفاده از قضیه بیز، با توجه به اینکه B رخ داده است، می توانیم احتمال وقوع A را پیدا کنیم. در اینجا، B شرط [۲۹] و A فرضیه [۳۰] است. فرضی که در اینجا مطرح می‌شود این است که پیش بینی کننده‌ها ویژگی‌ها مستقل هستند. یعنی وجود یک ویژگی خاص بر دیگری تأثیر نمی‌گذارد. از این رو به آن ساده [۳۱] گفته می‌شود [۱۸] .

$$P(A|B) = \frac{P(B|A) \cdot P(A)}{P(B)} \qquad (۱۲.۱)$$

[28] Naive Bayes Classifier
[29] Evidence
[30] Hypothesis
[31] Naive

پس اگر A را پارامتر هدف و B را بردار ویژگی در نظر بگیریم، داریم:

$$P(y|x_۱, \cdots, x_n) = \frac{P(x_۱|y) \cdots P(x_n|y)P(y)}{P(x_۱) \cdots p(x_n)} \qquad (۱۳.۱)$$

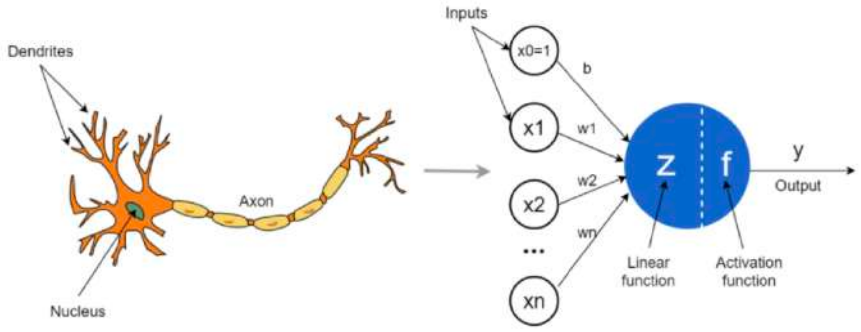
اکنون، می‌توانیم با مشاهده مجموعه داده‌ها، مقادیر هر یک را به دست آوریم و آن‌ها را در معادله جایگزین کنیم. برای همه ورودی‌های مجموعه داده، مخرج تغییر نمی‌کند. بنابراین می توان مخرج را حذف کرد و تناسبی را معرفی کرد.

$$P(y|x_۱, \cdots, x_n) \propto P(y) \prod_{i=۱}^{n} P(x_i|y) \qquad (۱۴.۱)$$

$$y = argmax_y P(y) \prod_{i=۱}^{n} P(x_i|y) \qquad (۱۵.۱)$$

## ۲.۱ شبکه‌های عصبی

شبکه‌های عصبی مصنوعی [۳۲] در حقیقت توسعه‌ای از مدل‌های شبه عصبی هستند که در دهه ۱۹۴۰ توسط دانشمندانی مانند وارن مک‌کالوک و والتر پیتس و برای حل مسائلی مانند تشخیص الگو و پردازش تصویر استفاده می‌شدند.
در دهه ۱۹۵۰، دانشمندانی مانند جان مک‌کارتی و فرانک روزنبلات، شروع به توسعه شبکه‌های عصبی بر اساس ایده‌های مدل‌های شبه عصبی کردند. اما در دهه ۱۹۶۰، با معرفی الگوریتم پس‌انتشار خطا [۳۳] توسط دانشمندانی مانند دیوید رملهارت و پائول ورز، شبکه‌های عصبی به مرحله بعدی توسعه و استفاده خود رسیدند.
در دهه‌های بعدی، با پیشرفت تکنولوژی و افزایش توان محاسباتی، شبکه‌های عصبی به عنوان یکی از قدرتمندترین روش‌های یادگیری ماشین شناخته شدند و در بسیاری از زمینه‌های کاربردی مانند تشخیص تصویر، ترجمه ماشینی، تشخیص گفتار و غیره، استفاده شدند.
بنابراین، شبکه‌های عصبی مصنوعی می‌توانند به عنوان یک نتیجه از پیشرفت در تحقیقات مدل‌های شبه عصبی در دهه‌های گذشته در نظر گرفته شوند [۱۹] .
هر شبکه عصبی مصنوعی ساخته شده از نورون‌هایی مصنوعی (مشابه نورون‌ها در سیستم عصبی انسان) است که از طریق اتصالاتی ( مشابه آکسون‌ها [۳۴] و دندریت‌ها [۳۵] ) با نورون‌های دیگر در ارتباط هستند. مانند سیناپس‌ها [۳۶] در سیستم عصبی انسان، هر ارتباط بین نورون‌ها سیگنال‌هایی را منتقل می‌کند که قدرت آن‌ها می‌تواند توسط وزنی که به طور مداوم در طول فرآیند یادگیری تنظیم می‌شود، افزایش یا کاهش یابد.




شکل ۱۱.۱: نورون مصنوعی و طبیعی

[32] Artificial Neural Network
[33] Backpropagation
[34] Axon
[35] Dendrite
[36] Synapse

### ۱.۲.۱ ساختار شبکه های عصبی

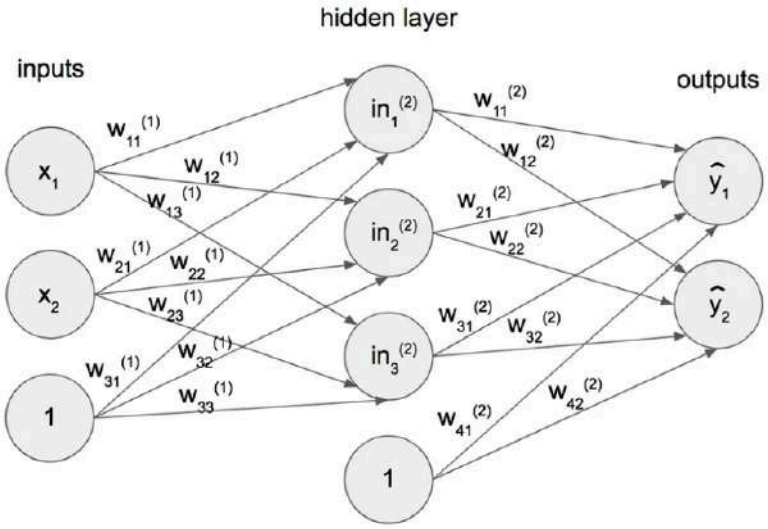
شبکه‌های عصبی مصنوعی می‌توانند به اشکال مختلفی ساخته شوند، اما بیشتر شامل چندین لایه از نورون‌ها هستند که به طور خاص به هم متصل شده‌اند. در ادامه، ساختار یک شبکه عصبی معمولی را توضیح می‌دهیم [۲۰] :

- **لایه ورودی** [۳۷] **:** این لایه شامل ورودی‌های شبکه است که به شکل عددی وارد شبکه می‌شوند. به طور معمول، هر نورون، یکی از ویژگی‌های مهم ورودی‌ها مانند رنگ، اندازه یا شکل را نشان می‌دهد.
- **لایه مخفی** [۳۸] **:** این لایه بین لایه ورودی و خروجی قرار دارد و شامل نورون‌هایی است که به طور مستقیم با لایه ورودی و خروجی ارتباط برقرار می‌کنند. هر نورون در این لایه، با ورودی‌های خود که از لایه قبلی وارد می‌شوند، محاسباتی را انجام می‌دهد و خروجی خود را به نورون‌های لایه بعدی می‌فرستد.
- **لایه خروجی** [۳۹] **:** این لایه شامل خروجی‌های شبکه است که به عنوان نتیجه شبکه به کاربر ارائه می‌شود. هر خروجی معمولاً یکی از کلاس‌های ممکن برای مسئله را نشان می‌دهد. برای مثال، در یک مسئله تشخیص تصویر، خروجی می‌تواند نشان دهنده یک شیء خاص در تصویر باشد.
- **وزن‌ها** [۴۰] **:** وزن‌ها نشان‌دهنده قدرت ارتباط بین نورون‌های مختلف در لایه‌های مختلف شبکه هستند. این وزن‌ها به طور مداوم در طول فرآیند یادگیری تنظیم می‌شوند تا بهترین نتیجه برای ورودی‌های جدید به دست آید.
- **تابع فعال‌سازی** [۴۱] **:** این تابع برای محاسبه خروجی نورون‌ها استفاده می‌شود. این تابع به عنوان تابعی غیرخطی عمل می‌کند تا شبکه بتواند الگوها و ویژگی‌های پیچیده‌تر را بیاموزد.
- **الگوریتم‌های یادگیری** [۴۲] **:** به منظور آموزش شبکه‌های عصبی، الگوریتم‌های یادگیری مانند الگوریتم پس‌انتشار خطا استفاده می‌شوند. این الگوریتم با استفاده از داده‌های آموزشی، وزن‌های شبکه را به طور مداوم تنظیم می‌کند تا بهترین نتیجه برای ورودی‌های جدید حاصل شود.

---

[37] Input Layer
[38] Hidden Layer
[39] Output Layer
[40] Weights
[41] Activation Function
[42] Learning Algorithms

- **اریبی‌ها** [۴۳] **:** اریبی‌ها به طور مشابه با وزن‌ها، پارامترهایی هستند که توسط شبکه به طور مداوم تنظیم می‌شوند. این پارامترها به عنوان مقداری ثابت در محاسبات هر نرون استفاده می‌شوند و به شبکه کمک می‌کنند تا بهترین نتیجه را به دست آورد.




شکل ۱۲.۱: ساختار شبکه عصبی

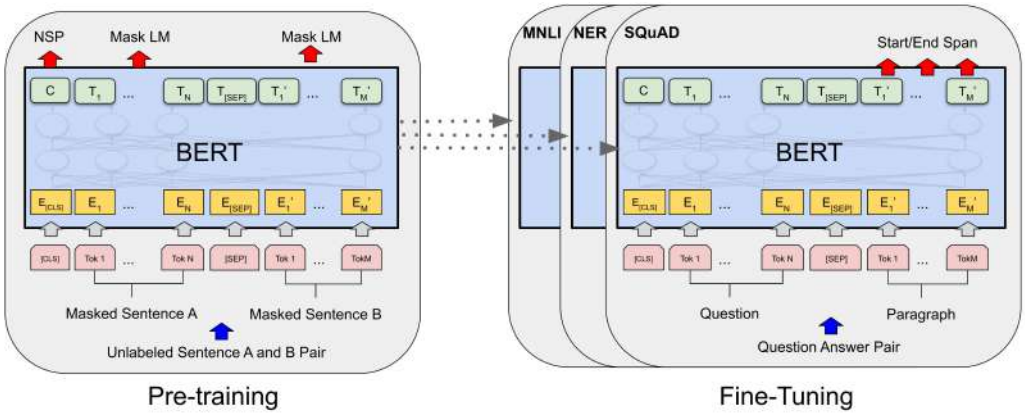
[43] Biases

## ۳.۱ شبکه‌های عصبی از پیش آموزش داده شده

شبکه‌های عصبی از پیش آموزش داده شده [۴۴] ، یکی از متدهای مهم در حوزه هوش مصنوعی هستند که در سال‌های اخیر بسیار مورد توجه قرار گرفته‌اند. این شبکه‌ها در ابتدا بر روی یک مجموعه داده آموزش داده می‌شوند و پس از آموزش، می‌توانند برای حل مسائل مشابه در زمینه‌های مختلف مورد استفاده قرار گیرند.
یکی از مزایای استفاده از شبکه‌های عصبی از پیش آموزش داده شده، کاهش زمان و هزینه آموزش است. آموزش یک شبکه عصبی بر روی یک مجموعه داده بزرگ و پیچیده، ممکن است مدت زمان زیادی را به خود اختصاص دهد و هزینه‌های بالایی را برای سازمان یا فرد آموزش دهنده به همراه داشته باشد. با استفاده از شبکه‌های عصبی از پیش آموزش داده شده، می‌توان زمان و هزینه آموزش را به شدت کاهش داد و در حل مسئله بهتر عمل کرد.
علاوه بر این، شبکه‌های عصبی از پیش آموزش داده شده، دارای قابلیت انتقال پذیری هستند. این به این معنی است که می‌توان از یک شبکه عصبی که برای حل یک مسئله خاص آموزش دیده است، برای حل مسائل مشابه در زمینه‌های مختلف استفاده کرد. به عنوان مثال، شبکه‌هایی که برای تشخیص تصویر آموزش دیده‌اند، می‌توانند برای حل مسائل مشابهی مانند تشخیص بخش‌های صورت، تشخیص اشیاء در تصاویر پزشکی و غیره استفاده شوند. به دلیل ویژگی‌های مشترکی که شبکه‌های عصبی از پیش آموزش داده شده با مسائل جدید دارند، می‌تواند عملکرد بهتری نسبت به شبکه‌های جدید داشته باشند.
برت [۴۵] یکی از مدل‌های شبکه‌های عصبی از پیش آموزش داده شده است که در حوزه پردازش زبان طبیعی [۴۶] استفاده می‌شود. این مدل توسط گوگل در سال ۲۰۱۸ معرفی شد و به سرعت محبوبیت بسیاری در میان پژوهشگران و صنعتگران به دست آورد.
برت در هسته‌ی خود دارای یک مدل زبانی مبتنی بر تبدیل‌کننده [۴۷]با تعداد زیادی از کدگذارها [۴۸]و لایه‌های self-attention است. برت از پیش روی دو مساله [۴۹] آموزش داده‌شده است. اولین مساله مدل‌کردن زبانی [۵۰] است که ۱۵ درصد از توکن‌ها را به صورت ماسک‌شده قرار داده بوده‌اند که برت آموزش داده شده‌است تا بر اساس متن آن‌ها را پیش‌بینی نماید. مساله دوم هم مرتبط با پیش‌بینی عبارت بعدی [۵۱] است. در این مساله برت آموزش داده شده است تا اگر یه جمله بعدی به صورت احتمالاتی داده شده باشد و یا اصلا جمله‌ی بعدی را نداشته باشیم چطور بتوانیم از روی یک جمله، جمله‌ی بعدی را پیش‌بینی نماییم. هردوی این مسئله‌ها، مسائل پایه‌ای و رایج در پردازش زبان‌های طبیعی هستند و در مسائل زیادی کاربرد دارند.

[44] Pre-trained Neural Networks
[45] BERT
[46] Natural Language Processing
[47] Transformer
[48] Encoders
[49] Task
[50] Language Modeling
[51] Next Sequence Prediction

به عنوان نتیجه می‌توان گفت که برت می‌تواند به صورت محتوایی کلمات را در جملات و متن‌های خاص یاد بگیرد. مزیت این مدل این است که پس از اینکه به صورت از پیش‌آموزش داده‌شده روی مساله‌های فوق آماده شد، می‌توان آن را روی هر مساله دلخواهی از پردازش زبان‌های طبیعی تنظیم [۵۲] کرد و امروزه با مجموعه داده‌های به نسبت کوچک روی مساله‌های دلخواه پردازش زبان‌های طبیعی هم به نتایج خوب با دقت‌های بالایی رسید و به صورت محاسباتی هم هزینه‌ی کمتری دارند چراکه در گذشته برت روی داده‌های زیادی آموزش داده شده‌است.




شکل ۱۳.۱: ساختار برت [۲۱]

[52] Fine-tune

## ۴.۱ تنظیم پارامتر

در مدل‌های یادگیری ماشین، پارامترها مقادیری هستند که توسط کاربر یا الگوریتم به مدل داده می‌شوند و می‌توانند بر روی عملکرد مدل تأثیر بگذارند. به‌عنوان مثال، در مدل‌های خطی مانند رگرسیون خطی و ماشین بردار پشتیبانی، پارامترهایی مانند ضرایب وزنی وجود دارند که تعیین می‌کنند که مدل چقدر قوی و عمومی خواهد بود. در مدل‌های دیگر مانند شبکه‌های عصبی، پارامترهایی مانند تعداد لایه‌ها، تعداد واحدهای هر لایه، نرخ یادگیری و نرخ کاهش گرادیان نیز وجود دارند که تعیین می‌کنند که مدل چقدر عمیق و پیچیده خواهد بود.
پیدا کردن پارامترهای بهینه می‌تواند عملکرد مدل را بهبود بخشد و تعیین پارامترهای نادرست می‌تواند منجر به بیش‌برازش یا کم‌برازش [۵۳] مدل شود که هر دو می‌توانند منجر به پایین آمدن دقت مدل بر روی داده‌های تست شوند [۲۲] .
برای تعیین پارامترهای مناسب در مدل‌های یادگیری ماشین، می‌توان از روش‌هایی مانند جستجوی خطی [۵۴] یا جستجوی تصادفی [۵۵] استفاده کرد. در هر یک از این روش‌ها، پارامترهای مختلفی برای مدل تعیین می‌شوند و برای هر یک از آن‌ها، مدل با پارامترهای داده شده روی داده‌های آموزشی، آموزش داده می‌شود و عملکرد آن روی داده‌های اعتبارسنجی بررسی می‌شود. سپس پارامترهایی که عملکرد بهتری را دارند، برای آزمون نهایی بر روی داده‌های تست استفاده می‌شوند.

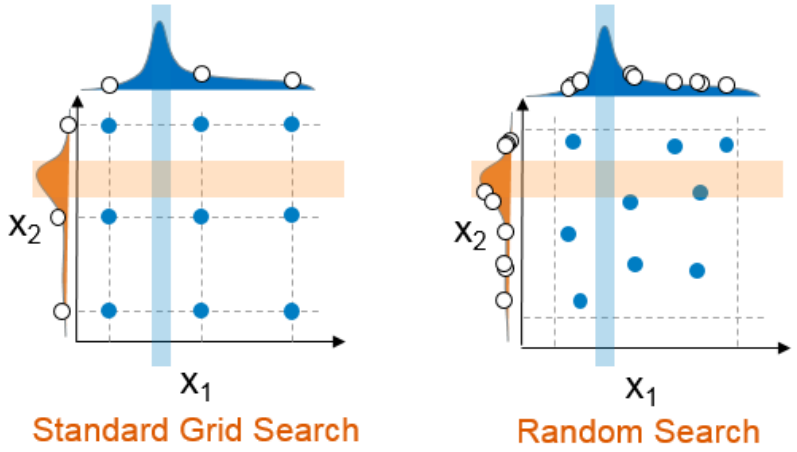
### ۱.۴.۱ جستجوی خطی

در جستجوی خطی، یک مجموعه مشخص از مقادیر برای هر ابرپارامتر انتخاب می‌شود و برای هر ترکیب ممکن از مقادیر ابرپارامترها، یک مدل آموزش داده می‌شود و عملکرد آن با استفاده از معیارهای ارزیابی مانند دقت [۵۶] ، فراخوانی [۵۷] و دقت پیش‌بینی [۵۸] ، ارزیابی می‌شود. سپس، مدلی که بهترین عملکرد را داشته باشد، به عنوان مدل نهایی انتخاب می‌شود. استفاده از جستجوی خطی می‌تواند به عنوان یک روش سریع و موثر برای پیدا کردن بهترین مقادیر ابرپارامترها در الگوریتم‌های یادگیری ماشین مورد استفاده قرار گیرد [۲۳] .

[53] Underfitting
[54] Grid Search
[55] Random Search
[56] Accuracy
[57] Recall
[58] Precision

### ۲.۴.۱ جستجوی تصادفی

جستجوی تصادفی نیز همانند جستجوی خطی، یک روش برای پیدا کردن بهترین مقادیر برای ابرپارامترهای یک الگوریتم یادگیری ماشین است. در این روش، به جای تعیین یک مجموعه مقادیر ثابت برای هر ابرپارامتر، از یک توزیع احتمالی برای هر پارامتر استفاده می‌شود. سپس، تعدادی ترکیب تصادفی از مقادیر ابرپارامترها انتخاب می‌شود و برای هر یک، یک مدل با استفاده از آن پارامترها آموزش داده می‌شود و عملکرد آن با استفاده از معیارهای ارزیابی، ، ارزیابی می‌شود. در نهایت، مدلی که بهترین عملکرد را داشته باشد، به عنوان مدل نهایی انتخاب می‌شود.



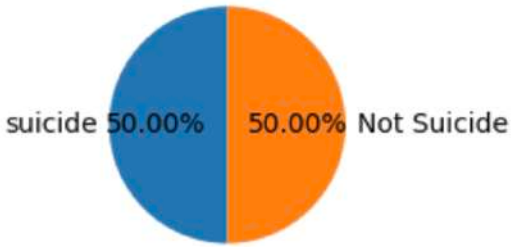

شکل ۱۴.۱: تفاوت جست و جوی خطی و تصادفی

# فصل ۲

# دادگان

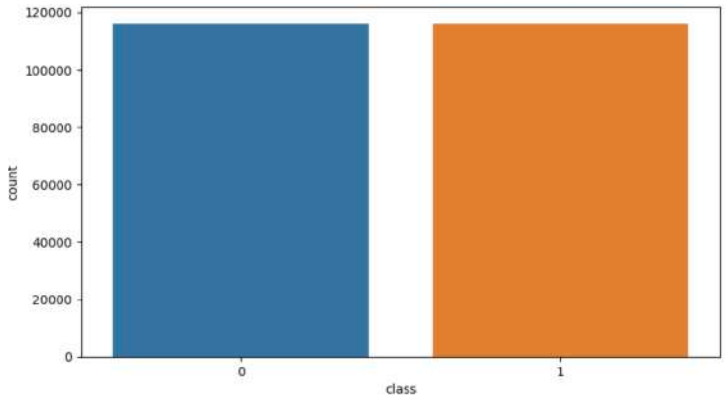
## ۱.۲ توصیف دادگان

این مجموعه داده دارای متونی است که توسط افراد در شبکه‌های اجتماعی، ایمیل‌های ارسال شده و پیام‌های متنی ارسال شده به دیگران نوشته شده است. این داده‌ها از سایت ردیت توسط API Pushshift جمع آوری شده‌اند. مجموعه داده شامل داده‌های سال‌های ۲۰۰۷ تا ۲۰۲۱ است. این داده‌ها می‌توانند برای تحلیل و پیش‌بینی رفتارهای آینده افراد مورد استفاده قرار گیرند و در کاهش نرخ خودکشی و بهبود سلامت روانی جامعه موثر باشند.



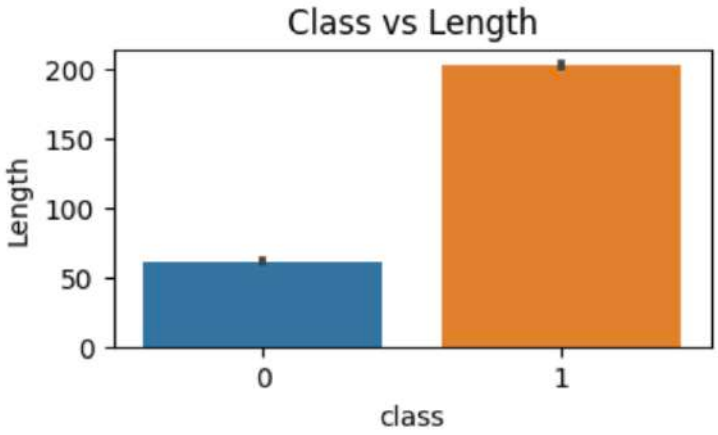

شکل ۱.۲: توزیع دادگان

در این مجموعه داده، داده‌ها در دو کلاس suicidal و non-suicidal طبقه‌بندی شده‌اند که به ترتیب متونی را دارا می‌باشند که شامل افرادی هستند که به نحوی درگیر خودکشی شده‌اند و افرادی که خودکشی نکرده‌اند. تعداد داده‌ها در هر کلاس با هم برابر است و این باعث می‌شود که داده‌ها همگن باشند.




شکل ۲.۲: تعداد داده‌های هر کلاس

این مجموعه داده شامل انواع مختلف متون است که در صورت استفاده مناسب می‌توانند به تحلیل‌های مفیدی درباره رفتارهای آینده افراد کمک کنند. همچنین طول متون متغیر است و شامل طول‌های مختلف از ۱ تا ۱۵۶۳۲ کاراکتر می‌باشد که اکثریت آن در بازه ۲۶ تا ۱۵۴ کاراکتر هستند.




شکل ۳.۲: طول داده‌های هر کلاس

## ۲.۲ پیش پردازش دادگان

**تبدیل تمام حروف متن به حروف کوچک:** این مرحله به منظور جلوگیری از در نظر گرفتن دو کلمه متفاوت به دلیل تفاوت حروف بزرگ و کوچک انجام شده است. برای مثال، کلمه Apple و apple به دلیل تفاوت حروف بزرگ و کوچک، به عنوان دو کلمه متفاوت در نظر گرفته خواهند شد. با تبدیل تمام حروف به حروف کوچک، این مشکل برطرف می‌شود و هر دو کلمه به عنوان یک کلمه در نظر گرفته می‌شوند.

**حذف فاصله ابتدایی و انتهایی متن:** در برخی موارد، ممکن است که دریافت داده‌ها از منابع مختلف و با کیفیت‌های مختلف انجام شود و این باعث می‌شود که در ابتدا یا انتهای متن، فاصله‌های اضافی وجود داشته باشد. این فاصله‌ها در پردازش متن ممکن است باعث ایجاد مشکلاتی شوند. به عنوان مثال، در برنامه‌های مربوط به پردازش زبان طبیعی، فاصله اضافی در ابتدا و انتهای متن ممکن است باعث ایجاد مشکلات در تشخیص یا جستجوی الگوها شود.

**حذف تگ‌های HTML:** در این مرحله، تمام تگ‌های HTML از متن حذف می‌شوند.حذف کد HTML و فرمت‌بندی از متن می‌تواند به کاهش حجم متن و بهبود قابلیت خواندن آن کمک کند.

**حذف همه علائم نگارشی از متن:** حذف علائم نگارشی می‌تواند بهبود قابلیت خواندن و پردازش متن را فراهم کند. علائم نگارشی ممکن است با کلمات دیگری به هم متصل شده و باعث ایجاد کلمات تکراری یا بدون معنی شوند. همچنین، در مواردی که متن باید برای پردازش به یک الگوریتم داده شود، حذف علائم نگارشی می‌تواند بهبود قابل توجهی در نتیجه نهایی ایجاد کند.

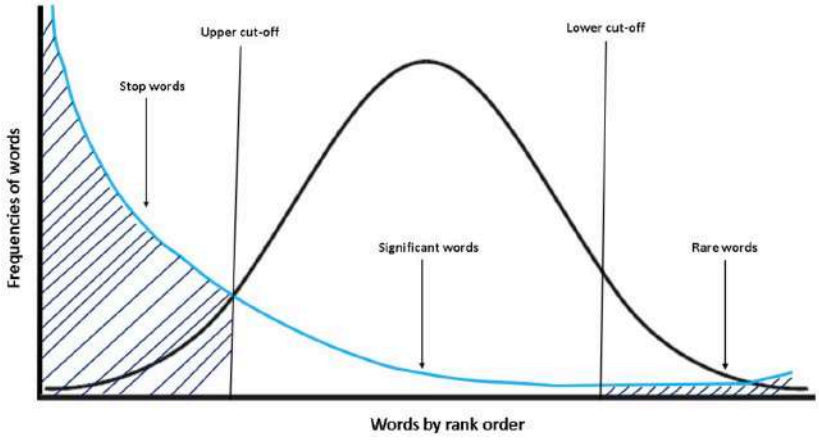
**جایگزینی هر نوع فاصله‌ای که بیش از یکی است :** در این مرحله، تمام فاصله‌های اضافی بین کلمات را حذف کرده و یک فاصله به عنوان جایگزین آن‌ها قرار می‌دهیم. این کار به منظور بهبود کیفیت پردازش متن و حذف هرگونه فاصله اضافی بین کلمات است.

**جایگزینی کردن هر کاراکتر غیر الفبایی :** در این مرحله، تمام کاراکترهای غیر الفبایی مانند اعداد، نمادهای خاص، علائم ریاضی و... با یک فاصله جایگزین می‌شوند.

**حذف ایست واژه:** طبق قانون Zipf ، اگر نمودار بین فراوانی توزیع کلمه و رتبه یک کلمه را در یک متن رسم کنیم، متوجه خواهیم شد که بسامد کلمات با رتبه یک کلمه نسبت معکوس دارد، یعنی رتبه ۱ به پر تکرار ترین کلمه، رتبه ۲ به دومین کلمه پرتکرار و ... داده می‌شود.

از نمودار ۴.۲ می توان نتیجه گرفت که به طور کلی سه نوع کلمه وجود دارد:

- کلمات توقف: این دسته کلمات درواقع کلمات پر تکرار مانند a ، an ، the ، is و ... هستند.
- کلمات مهم: آن دسته از کلمات هستند که دارای فراوانی متوسطی هستند و معنای واقعی را به متن اضافه می‌کنند. این کلمات مهم تر از کلمات توقف هستند.
- کلمات نادر: کلماتی هستند که با فراوانی بسیار کمتری به وجود می‌آیند و اهمیت نسبتاً کمتری نسبت به کلمات مهم دارند. کلماتی که به ندرت وجود دارند ممکن است برای درک متن مفید باشند یا نباشند.




شکل ۴.۲: نمودار بسامد - رتبه

کلمات توقف به دو دلیل از متن حذف می‌شوند:

- اطلاعات مهمی به متن اضافه نمی‌کنند.
- فراوانی کلمات توقف در یک متن بسیار زیاد است. حذف این کلمات از متن، باعث کوچک شدن حجم داده‌ها و افزایش سرعت محاسبات می‌شود.

**ریشه‌یابی کلمات:** [1] فرآیند الگوریتمی یافتن لم یک کلمه بسته به معنی کلمه و محل قرارگیری آن در جمله است. برای مثال ریشه‌ی کلمه‌ی cats ، cat است و ریشه‌ی کلمه‌ی running ، run است. ریشه یابی کلمات مزایای زیادی دارد. از جمله آن‌ها عبارتند از:

- دقت تحلیل متن را بهبود می بخشد: برگرداندن کلمات، باعث می‌شود کلمات هم معنی راحت‌تر شناسایی شوند.
- اندازه داده را کاهش می دهد: از آنجایی که کلمات را به شکل پایه‌شان تبدیل می‌کند، باعث کاهش اندازه داده متن می‌شود
- نتایج جستجوی بهتر: از آنجایی که اشکال مختلف یک کلمه همه به یک فرم تبدیل می‌شوند ، جست و جو راحت تر و با دقت بهتری انجام می‌شود.

**تعیین نقش کلمات:** برای ریشه‌یابی کلمات، معنی آن‌ها مورد نیاز است و در پیدا کردن معنی کلمات، نقش آن‌ها در جمله پارامتر مهمی است که تابع ریشه‌یاب از آن استفاده می‌کند. از آنجایی که ما برای ریشه‌یابی کلمات از تابع WordNetLemmatizer استفاده کردیم، نیازمند این هستیم که نقش کلمات را در فرم WordNet داشته باشیم. پس با استفاده از دستور wordnet.POS این کار را انجام می‌دهیم [۲۴] .

**تبدیل کلمات به بردار:** [2] راه‌های متفاوتی برای بردار سازی کلمات وجود دارد. در این مقاله از روش‌های زیر استفاده شده است:

- CountVectorizer: این تابع توسط کتابخانه scikit-learn پیاده سازی شده است. این تابع مراحل زیر را طی می‌کند:
  - لیستی از رشته‌ها دریافت می‌کند.
  - هر کدام از رشته‌ها را به توکن‌هایشان تقسیم‌بندی می‌کند.
  - یک لیست از توکن‌های یکتای کل ورودی‌ها پیدا می‌کند. عضوهای این لیست، اعضای بردار ویژگی هستند.
  - با استفاده از یک حلقه، روی تمام ورودی‌ها حرکت می‌کند و تعداد تکرار هر کدام از ویژگی‌ها در متن را پیدا می‌کند.
  - یک ماتریس اسپارس از ورودی می‌سازد و خروجی تابع همین ماتریس است.

[1] Lemmatization
[2] Vectorization

| | about | bird | heard | is | the | word | you |
|---|---|---|---|---|---|---|---|
| About the bird, the bird, bird bird bird | 1 | 5 | 0 | 0 | 2 | 0 | 0 |
| You heard about the bird | 1 | 1 | 1 | 0 | 1 | 0 | 1 |
| The bird is the word | 0 | 1 | 0 | 1 | 2 | 1 | 0 |

شکل ۵.۲: تبدیل متن به بردارهای عددی

- TfidfVectorizer: این تابع نیز توسط کتابخانه scikit-learn پیاده سازی شده و مراحل آن به شرح زیر است:
  - رشته‌های ورودی را به توکن تقسیم می‌کند.
  - تعداد تکرار هر توکن در هر رشته را پیدا می‌کند (TF)
  - برای هر کلمه فرکانس معکوس (IDF) را محاسبه می‌کند. فرکانس معکوس معیاریست از میزان رایج یا نادر بودن کلمه در ورودی.
  - سپس امتیاز TF-IDF را برای هر کلمه در هر ورودی محاسبه می‌کند. این امتیاز به شکل زیر محاسبه می‌شود.

$$TF(i,j) = \frac{\text{Term i frequency in document j}}{\text{Total words in document j}} \tag{2.1}$$

$$IDF(i) = \log_2 \frac{\text{Total documents}}{\text{documents with term j}} \tag{2.2}$$

$$score = TF \cdot IDF \tag{2.3}$$

  - با استفاده از امتیازهای بدست آمده، ماتریس TF-IDF را می‌سازد [۲۵] .

در تابع پیاده‌سازی شده در scikit-learn پارامترهایی برای ایجاد تغییر در ماتریس خروجی وجود دارد. در این مقاله، از دو مورد از این پارامترها استفاده شده است. این پارامترها عبارتند از max features و ngram range. پارامتر اول مربوط به تعداد سطون‌های ماتریس خروجی است. در این پروژه این پارامتر برابر با ۱۰۰۰ قرار داده شده به این معنا که ۱۰۰۰ کلمه‌ی پر تکرار مورد بررسی قرار می‌گیرند. پارامتر دوم مربوط به تعداد بخش‌های کلمه است. در این پروژه این پارامتر را با دو مقدار استفاده کردیم.

| Documents | Text | Total number of words in a document |
|---|---|---|
| A | Jupiter is the largest planet | 5 |
| B | Mars is the fourth planet from the sun | 8 |

(آ) جملات

| Words | TF (for A) | TF (for B) | IDF | TFIDF (A) | TFIDF (B) |
|---|---|---|---|---|---|
| Jupiter | 1/5 | 0 | ln(2/1) = 0.69 | 0.138 | 0 |
| Is | 1/5 | 1/8 | ln(2/2) = 0 | 0 | 0 |
| The | 1/5 | 2/8 | ln(2/2) = 0 | 0 | 0 |
| largest | 1/5 | 0 | ln(2/1) = 0.69 | 0.138 | 0 |
| Planet | 1/5 | 1/8 | ln(2/2) = 0 | 0.138 | 0 |
| Mars | 0 | 1/8 | ln(2/1) = 0.69 | 0 | 0.086 |
| Fourth | 0 | 1/8 | ln(2/1) = 0.69 | 0 | 0.086 |
| From | 0 | 1/8 | ln(2/1) = 0.69 | 0 | 0.086 |
| Sun | 0 | 1/8 | ln(2/1) = 0.69 | 0 | 0.086 |

(ب) بردارسازی

شکل ۶.۲: TF-IDF

- مقدار پیش فرض آن که فقط کلمات تک بخشی مانند machine و subfield را به عنوان ویژگی در نظر میگیرد.
- آن را برابر با (۱, ۲) قرار دادیم که باعث می‌شود کلمات دو بخشی مانند machine learning و blue bottle هم در نظر گرفته می‌شوند.

- تخصیص نمرات احساسات به متن با استفاده از مدل تحلیل احساسات VADER. مدل تحلیل احساسات VADER با توجه ویژگی‌های هر کلمه، به آن نمره‌ای بین یک (معنای مثبت) و منفی یک (معنای منفی) نسبت می‌دهد. این ویژگی‌ها عبارتند از [۲۶] :
  - کلمات:
    Good = + 0.8, Bad = - 0.8
  - کلمات خاص نشان دهنده وضعیت [۳] :
    amazing = +1.0, horrible = -0.9

[3] Amplifiers

- اموجی‌ها [۴] :
باتوجه به بار معنایی هر اموجی به آن یک عدد نسبت می‌دهد.
- ترکیب‌ها [۵] : sort of = -0.3

همچنین روابطی مبتنی بر احساسات [۶] نیز دارد. مانند:

- با حروف بزرگ نوشتن:
SO GOOD = +0.9
- کلماتی که نشان دهنده‌ی میزان احساست هستند:
very good = +0.9, insanely good = +1.0
- کلمات منفی یا مثبت کننده [۷] :
not good = -0.8

با استفاده از این مدل، به هر سطر از داده چهار عدد نسبت داده می‌شود.

| Input | neg | neu | pos | compound |
|---|---|---|---|---|
| "This computer is a good deal." | 0 | 0.58 | 0.42 | 0.44 |
| "This computer is a very good deal." | 0 | 0.61 | 0.39 | 0.49 |
| "This computer is a very good deal!!" | 0 | 0.57 | 0.43 | 0.58 |
| This computer is a very good deal!! :-)" | 0 | 0.44 | 0.56 | 0.74 |
| This computer is a VERY good deal!! :-)" | 0 | 0.393 | 0.61 | 0.82 |

شکل ۷.۲: امتیازات VADER

[4] Emojies
[5] Combos
[6] Sentiment-Based
[7] Negation Reversal

## ۳.۲ تقسیم‌بندی دادگان

برای آماده‌سازی داده‌ها برای آموزش، الگوریتم‌های یادگیری ماشین، نخست باید داده‌ها را به بخش‌های آموزش، تست و اعتبارسنجی تقسیم کنیم. در بخش آموزش الگوریتم داده‌ها را می‌بیند و مدل را می‌آموزد، سپس با استفاده از داده‌های اعتبار سنجی پارامترهای مدل را تنظیم و در بخش تست مدل را بر روی داده‌ای که هیچ‌گاه ندیده بررسی می‌کند.
۳۰ درصد از داده‌ها به عنوان داده تست از مجموعه داده کنار گذاشته شده و از داده‌های باقی‌مانده ۲۰ درصد به عنوان داده اعتبار سنجی و ۸۰ درصد به عنوان داده آموزشی در نظر گرفته شده‌اند.
برای تقسیم بندی داده‌ها از تابع train test split از کتابخانه scikit-learn استفاده شده است.
مهم‌تر از همه اینکه قبل از تقسیم، تمامی داده‌ها را بر می‌زنیم [۸] . این کار جلوی اثرات احتمالی بعضی از سطرهای داده را می‌گیرد و عملکرد مدل را بهبود می‌بخشد.

[8] Shuffle

# فصل ۳

# آزمایشات و نتایج

در این بخش به بررسی نتایج بدست آمده توسط هر یک از الگوریتم‌های ذکر شده در فصل یک می‌پردازیم.
می‌دانیم که داده‌های پروژه را به چهار روش مختلف تبدیل به بردار کردیم (بخش ۲.۲). در هر الگوریتم ابتدا برای هرکدام از این چهار روش، با استفاده از جست و جوی خطی، بهترین پارامترها را پیدا کرده و سپس برای هر روش، مدل را با پارامترهای ایده‌آل آموزش می‌دهیم. سپس دقت مدل را با معیارهای متفاوتی حساب می‌کنیم. برای توضیح معیارها ابتدا نیاز به نمادگذاری داریم.
اعضای کلاس ۱ که درست برآورد شده‌اند: True Positive
اعضای کلاس صفر که درست برآورد شده‌اند: True Negative
اعضای کلاس ۱ که اشتباه برآورد شده‌اند: False Negative
اعضای کلاس صفر که اشتباه برآورد شده‌اند: False Positive
حال با استفاده از این مقادیر، ماتریس درهم‌ریختگی [1] را رسم می‌کنیم. این ماتریس به شکل زیر بدست می‌آید:

$$\begin{bmatrix} TruePositive & FalsePositive \\ FalseNegative & TrueNegative \end{bmatrix}$$

سپس معیارهای دقت را بدست می‌آوریم. این معیارها عبارتند از [۲۷] :

- معیار صحت :

$$Precision = \frac{TP}{TP + FP} \qquad (۱.۳)$$

[1] Confusion Matrix

- معیار پوشش :

$$Recall = \frac{TP}{TP + FN} \qquad (۲.۳)$$

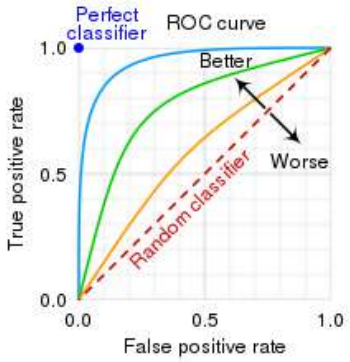
- امتیاز F1:

$$F1 = 2 \cdot \frac{Precision \cdot Recall}{Precision + Recall} \qquad (3.3)$$

- دقت :

$$Accuracy = \frac{TP + TN}{TP + FP + TN + FN} \qquad (۴.۳)$$

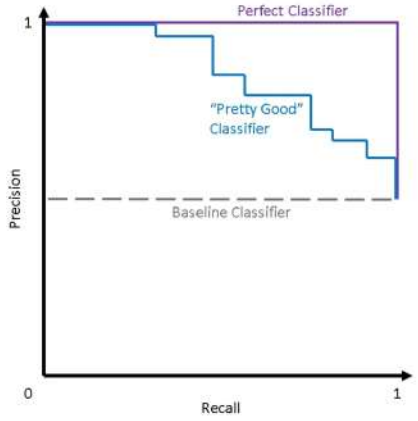
پس از بدست آوردن معیارها، آن‌ها را در نموداری میله‌ای نمایش می‌دهیم.
نمودار بعدی که رسم می‌کنیم، نمودار **مشخصه عملکرد** [۲] است [۲۸] . این نمودار ترسیم نسبت نرخ مثبت صحیح [۳] بر حسب نرخ مثبت کاذب [۴] است. بهترین عملکرد دسته‌بندی در این نمودار در نقطه‌ای با مختصات $(۰, ۱)$ رخ خواهد داد. در این نقطه کمترین نرخ اشتباه و بیشترین نرخ صحیح را داریم. هر چه داده‌های دو کلاس به هم شبیه‌تر باشند و تمایز آن‌ها سخت‌تر باشد، منحنی این نمودار به خط $y = x$ نزدیک‌تر است.




شکل ۱.۳: نمودار مشخصه عملکرد

[2]Receiver Operating Characteristic
[3]True Positive Rate
[4]False Positive Rate

آخرین نموداری که رسم می‌کنیم، نمودار **صحت-پوشش** [5] است. الگوریتم طبقه‌بندی خوب است که هم صحت و هم پوشش بالایی داشته باشد. پس با استفاده از منحنی این نمودار می‌توانیم میزان خوب بودن یک الگوریتم طبقه‌بندی را بفهمیم.



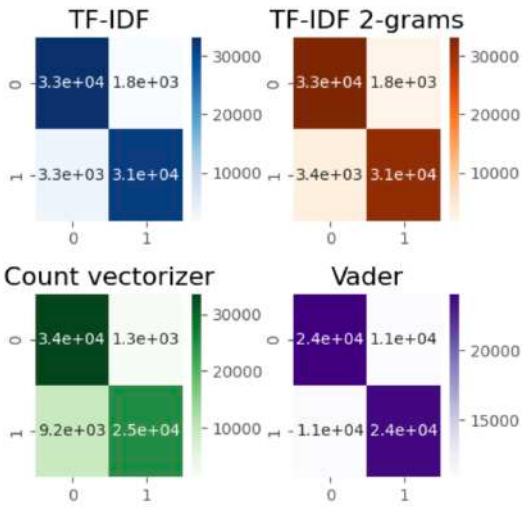

شکل ۲.۳: نمودار صحت-پوشش

# ۱.۳ رگرسیون

## ۱.۱.۳ رگرسیون خطی ساده

نتایج آزمایش رگرسیون خطی در شکل ۴.۳ نمایش داده شده است. با توجه به ماتریس درهم ریختگی، مدل‌هایی که با داده‌های TF-IDF و TF-IDF(n=(1,2)) آموزش داده شده‌اند، شبیه به هم بوده و تعداد زیادی از داده‌ها را درست پیش‌بینی کرده‌اند.
مدلی که با داده‌های Count Vectorized آموزش دیده، در پیش‌بینی داده‌های کلاس صفر بهتر از دو مدل قبلی عمل کرده اما عملکرد کلی آن به خوبی آن‌ها نیست.
مدل آموزش داده شده با داده‌های Vader کم دقت‌ترین مدل بوده و نتایج خوبی نداشته است.

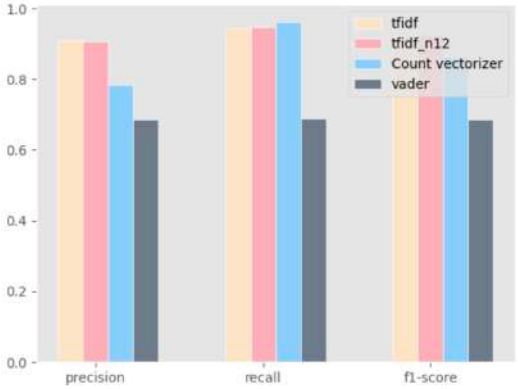
[5] Precision-Recall Curve




شکل ۳.۳: ماتریس درهم ریختگی الگوریتم رگرسیون خطی

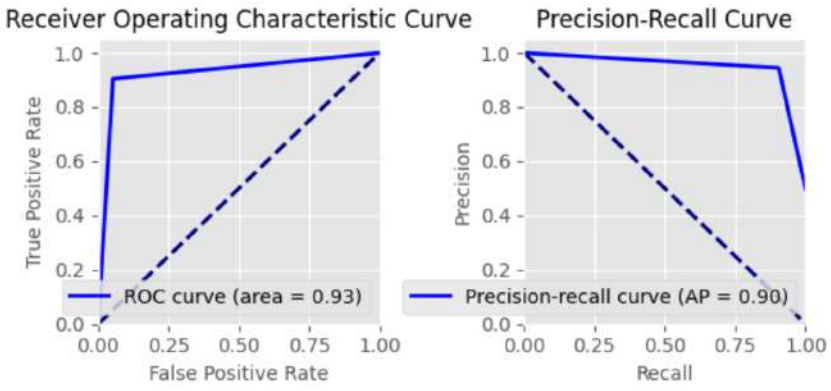
معیارهای دقت را در جدول زیر می‌توانید مشاهده کنید:

| metric | TF-IDF | TF-IDF 2 | CountVec | Vader |
|---|---|---|---|---|
| precision | 0.908 | 0.906 | 0.784 | 0.685 |
| recall | 0.947 | 0.948 | 0.962 | 0.688 |
| F1-score | 0.927 | 0.927 | 0.864 | 0.687 |
| accuracy | 0.925 | 0.925 | 0.848 | 0.685 |

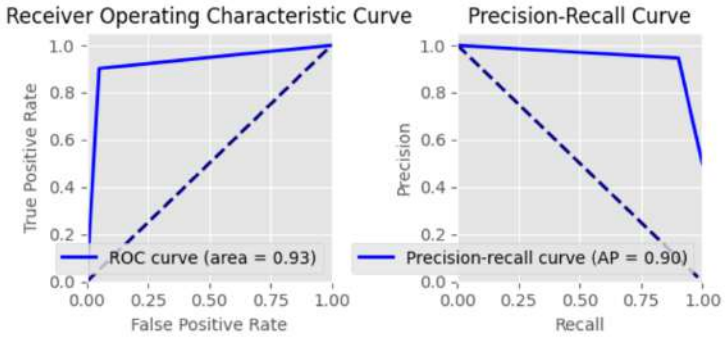



شکل ۴.۳: نتایج الگوریتم رگرسیون خطی

حال به رسم نمودارهای مشخصه عملکرد و صحت_پوشش می‌پردازیم.

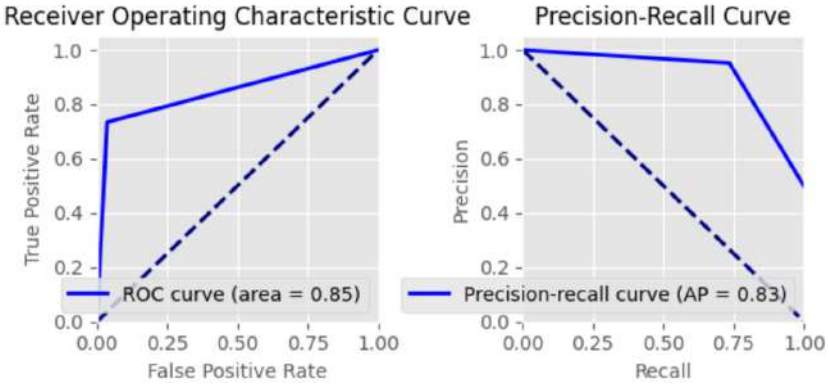



TF-IDF (آ)

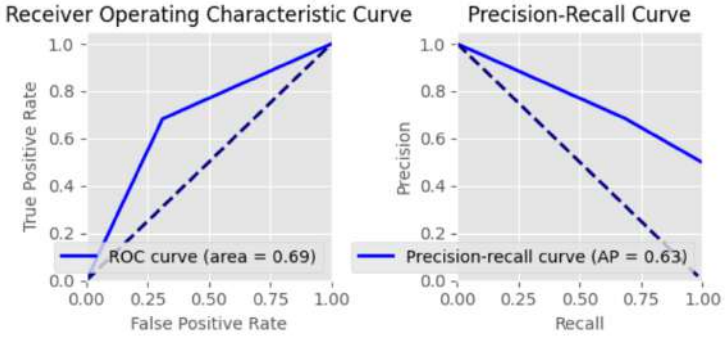



TF-IDF - ۲ (ب)

شکل ۵.۳: نمودارهای مشخصه عملکرد و صحت پوشش رگرسیون خطی




CountVec (آ)




Vader (ب)

شکل ۶.۳: نمودارهای مشخصه عملکرد و صحت پوشش رگرسیون خطی

همانطور که از مساحت زیر نمودارها مشخص است، داده‌های TF-IDF با دارا بودن ۰/۹۳ و ۰/۹۰ مساحت زیر نمودارهای مشخصه عملکرد و صحت پوشش، بهترین نتایج و بعد از آن‌ها داده‌های Count vectorization با ۰/۸۵ و ۰/۸۳ در رتبه بعدی و در نهایت هم داده‌های Vader با ۰/۶۹ و ۰/۶۳ در جایگاه آخر قرار دارند.

### ۲.۱.۳ رگرسیون لجستیک

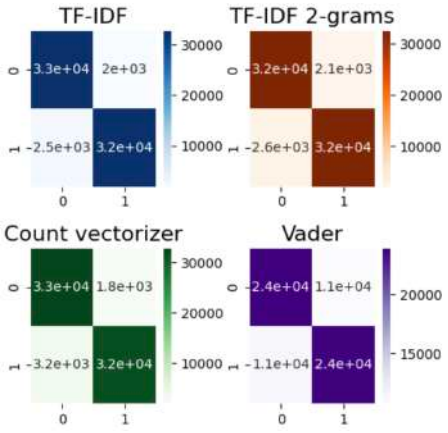
پارامترهایی که در این مدل تلاش به بهینه کردن آن‌ها می‌کنیم عبارتند از:

- c: معکوس قدرت منظم سازی [۶] است. برای کنترل مبادله بین دستیابی به یک خطای آموزشی کم و یک خطای تست کم استفاده می‌شود. هرچه مقدار این پارامتر کمتر باشد، قدرت منظم‌سازی بیشتر و در نتیجه مدل ساده‌تر است. مدل ساده‌تر قدرت تعمیم بیشتری دارد اما خطای بیشتری روی داده‌ی آموزشی دارد. همچنین زیاد بودن مقدار این پارامتر می‌تواند مدل پیچیده‌ای تولید کند که داده آموزشی را حفظ می‌کند.
- Penalty: مدل تابع منظم ساز را مشخص می‌کند. می‌تواند L1 یا L2 باشد.

پارامتر های بدست آمده توسط جست و جوی خطی برای رگرسیون لجستیک به شرح زیر است:

| Method | c | penalty |
|---|---|---|
| TF-IDF | 10 | l2 |
| TF-IDF(n=(1,2)) | 10 | l2 |
| CountVectorizer | 1 | l2 |
| Vader(emotion) | 0.1 | l2 |

با استفاده از پارامترهای بدست آمده مدل را آموزش دادیم. ماتریس درهم ریختگی در شکل ۷.۳ رسم شده است.

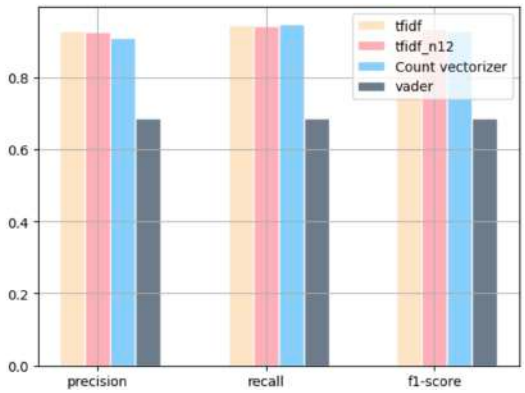



شکل ۷.۳: ماتریس درهم ریختگی الگوریتم رگرسیون لجستیک

[6] Regularization

با توجه به ماتریس، تعداد TP و TN به طور کلی بیشتر از تعداد FP و FN است که نشانه خوبی است. این نشان می‌دهد که رگرسیون لجستیک موارد مثبت و منفی را با دقتی معقول شناسایی می‌کند. با مقایسه چهار ماتریس درهم ریختگی، به نظر می‌رسد که سه ماتریس اول (TF-IDF, TF-IDF-2, Count Vectorize) ، عملکرد مشابهی دارند، با تعداد بالایی مثبت و منفی واقعی، و تعداد نسبتاً کم مثبت و منفی نادرست.این نتایج نشان می‌دهد که مدل رگرسیون لجستیک هنگام استفاده از این سه روش بردارسازی، عملکرد خوبی دارد. از سوی دیگر، چهارمین ماتریس درهم ریختگی (vader) در مقایسه با سه ماتریس دیگر، تعداد کمتری از مثبت‌ها و منفی‌های درست را نشان می‌دهد. این نشان می‌دهد که این روش بردارسازی به اندازه سه روش دیگر برای این کلاس‌بندی مؤثر نبوده است.
معیارهای دقت را در جدول زیر می‌توانید مشاهده کنید.

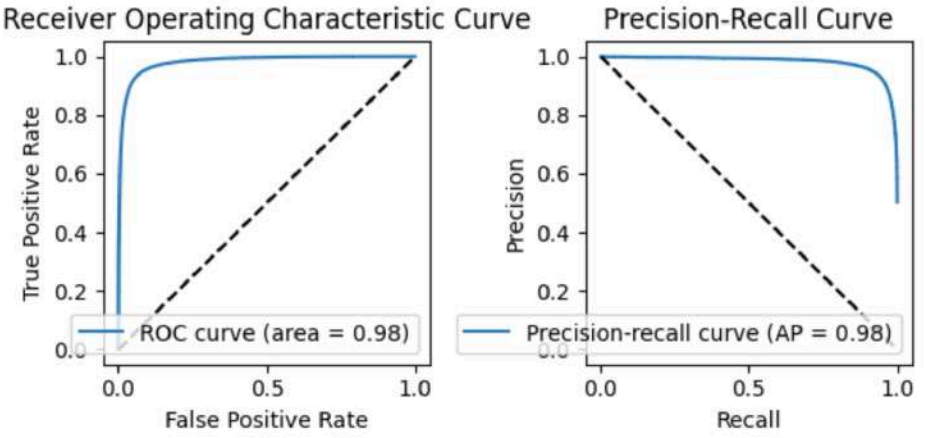
| metric | TF-IDF | TF-IDF 2 | CountVec | Vader |
|---|---|---|---|---|
| precision | 0.929 | 0.926 | 0.910 | 0.684 |
| recall | 0.944 | 0.942 | 0.948 | 0.686 |
| F1-score | 0.936 | 0.934 | 0.928 | 0.685 |
| accuracy | 0.936 | 0.933 | 0.927 | 0.685 |



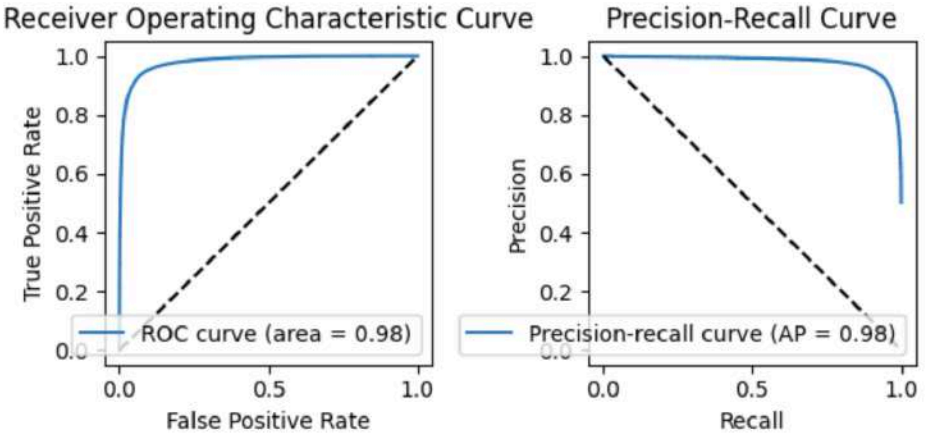

شکل ۸.۳: نمودار نتایج رگرسیون لجستیک

این نتایج نشان می‌دهد که با الگوریتم رگرسیون لوجستیک، روش‌های بردار سازی TF-IDF و TF-IDF-2 موثرتر از روش‌های CountVec و Vader در این مسئله عمل کردند.

حال به رسم نمودارهای مشخصه عملکرد و صحت_پوشش می‌پردازیم.

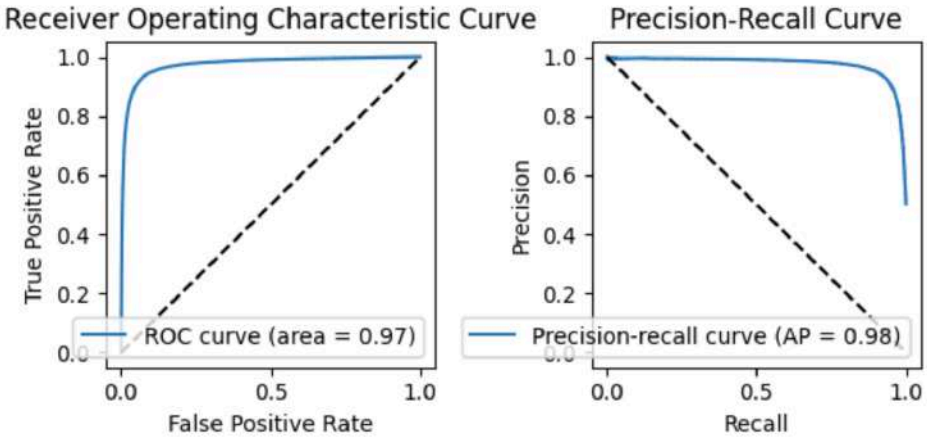



TF-IDF (آ)

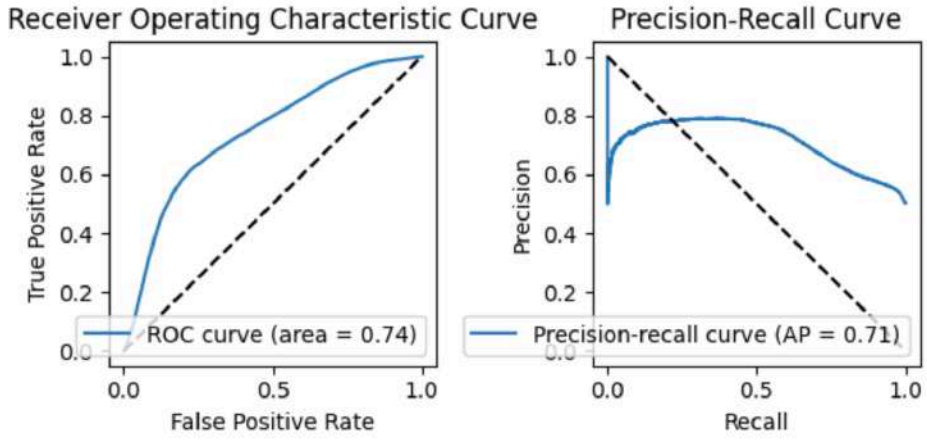



TF-IDF - ۲ (ب)

شکل ۹.۳: نمودارهای مشخصه عملکرد و صحت پوشش رگرسیون لجستیک




CountVec (آ)




Vader (ب)

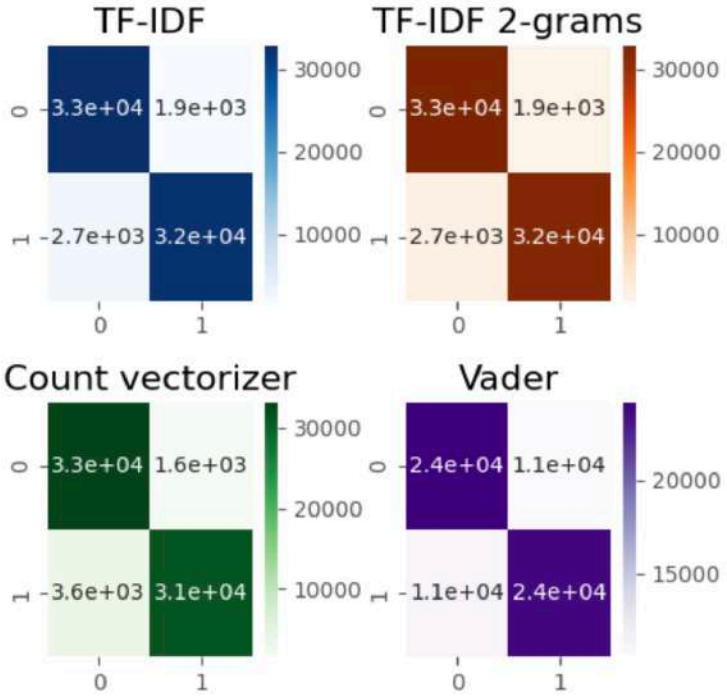
شکل ۱۰.۳: نمودارهای مشخصه عملکرد و صحت پوشش رگرسیون لجستیک

همانطور که از شکل ۷.۳ هم مشخص بود، با توجه به اینکه در جفت نمودارهای شکل ۹.۳ و بخش (آ) شکل ۱۰.۳ نمودار به نقطه‌ی بهینه بسیار نزدیک است ، می‌توان نتیجه گرفت که الگوریتم نتیجه خوبی داشته. اما در نمودار (ب) شکل ۱۰.۳ می‌توان دید که نمودار از نقاط بهینه دور است و خوب عمل نکرده است.

## ۲.۳ طبقه بند بردار پشتیبان خطی

با توجه به مجموعه داده‌های پروژه و همچنین توضیحات ذکر شده در فصل اول، به این نتیجه می‌رسیم که در این مسئله الگوریتم های رگرسیون لجستیک و بردار پشتیبان مثل یکدیگر عمل کرده و در یک صفحه‌ی دو بعدی(صفحه‌ی x-y ) منحنی‌های خطی و غیر خطی را برازش می‌کنند. پس نتایج این دو الگوریتم یکی خواهد بود. این موضوع از نتایج بردار پشتیبان خطی که در جدول زیر نمایش داده شده است نیز مبرهن است:

| metric | TF-IDF | TF-IDF 2 | CountVec | Vader |
|---|---|---|---|---|
| precision | 0.926 | 0.926 | 0.891 | 0.681 |
| recall | 0.944 | 0.902 | 0.956 | 0.692 |
| F1-score | 0.935 | 0.934 | 0.923 | 0.687 |
| accuracy | 0.0.935 | 0.934 | 0.920 | 0.686 |




شکل ۱۱.۳: بردار درهم ریختگی الگوریتم بردار پشتیبان

## ۳.۳ الگوریتم جنگل تصادفی

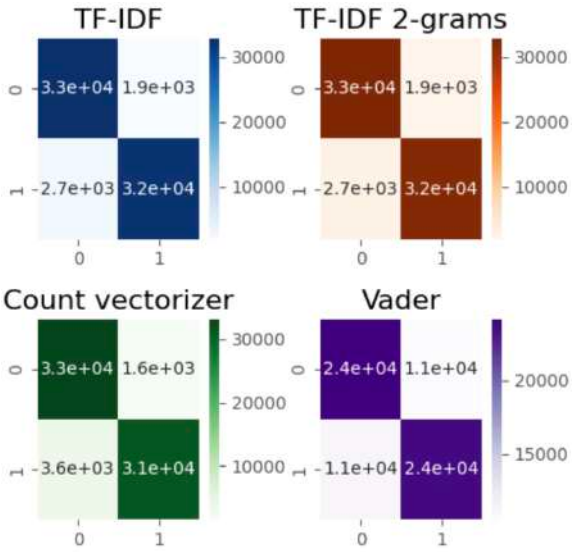
پارامترهای تنظیم شده برای این الگوریتم عبارتند از:

- max depth : این پارامتر نشان دهنده‌ی حداکثر عمق هرکدام از درخت‌های تصمیم در جنگل تصادفی است. بیشتر بودن این پارامتر به معنای درخت‌های عمیق‌تر و در نتیجه مدل پیچیده‌تر است.
- min sample split : حداقل تعداد نمونه‌های لازم برای تقسیم کردن یک راس از درخت تصمیم را نشان می‌دهد. بیشتر کردن مقدار این پارامتر باعث می‌شود درخت با تعداد کم داده تقسیم نشود و می‌تواند جلوی بیش برازش مدل را بگیرد.
- min sample leaf : این پارامتر مینیمم تعداد داده لازم برای تشکیل یک راس از درخت را نشان می‌دهد. هر چه مقدار این پارامتر بیشتر باشد، داده بیشتری برای تشکیل یک راس لازم بوده و احتمال بیش‌برازش کمتر است.
- n estimator : تعداد درخت‌های تصمیم موجود در جنگل تصادفی را نشان می‌دهد. تعداد بیشتر درخت‌ها باعث می‌شود زمان بیشتری برای آموزش صرف شود ولی می‌تواند دقت مدل را افزایش بدهد.

پارامترهای بهینه بدست آمده برای این الگوریتم عبارتند از:

| Method | min samples leaf | min samples split | n estimators |
|---|---|---|---|
| TF-IDF | 1 | 5 | 200 |
| TF-IDF(n=(1,2)) | 1 | 5 | 100 |
| CountVectorizer | 1 | 2 | 200 |
| Vader(emotion) | 2 | 5 | 50 |

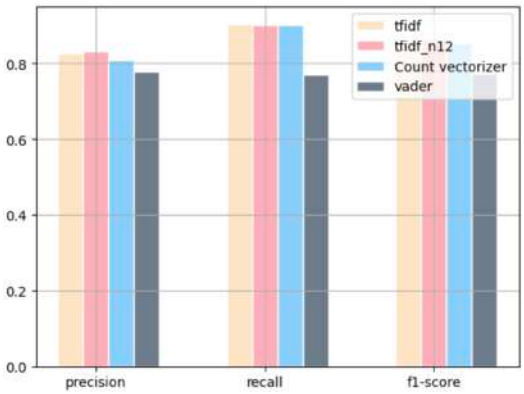
با اعمال پارامترهای فوق و آموزش دادن مدل به نتایج زیر می‌رسیم:




شکل ۱۲.۳: بردار درهم ریختگی الگوریتم جنگل تصادفی

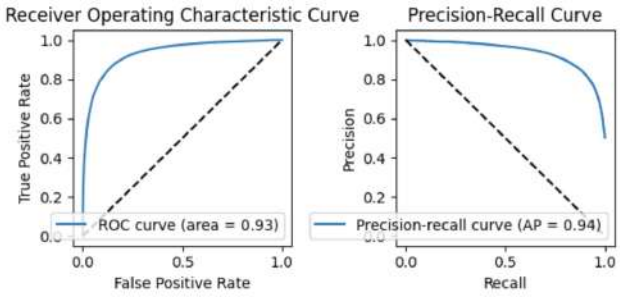
با توجه به بردار درهم ریختگی شکل ۱۲.۳ متوجه می‌شویم که این الگوریتم نیز مانند الگوریتم‌های قبلی، روی داده‌های TF-IDF و TF-IDF-2 و Count Vectorizer نتیجه بهتری نسبت به داده‌های Vader دارد.

| metric | TF-IDF | TF-IDF 2 | CountVec | Vader |
|---|---|---|---|---|
| precision | 0.827 | 0.831 | 0.807 | 0.777 |
| recall | 0.904 | 0.902 | 0.900 | 0.769 |
| F1-score | 0.864 | 0.865 | 0.851 | 0.773 |
| accuracy | 0.857 | 0.859 | 0.843 | 0.774 |

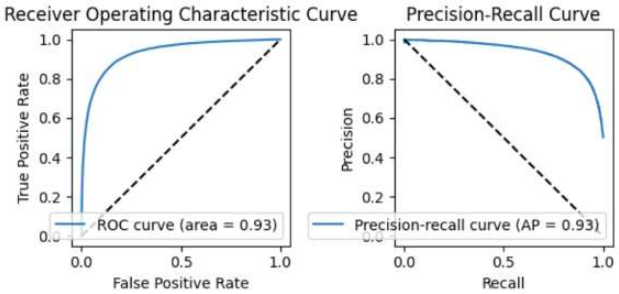



شکل ۱۳.۳: نمودار نتایج جنگل تصادفی

با توجه به نتایج بردار درهم ریختگی و نتایج بدست آمده، انتظار می‌رود نمودارهای مشخصه‌ـ عملکرد و صحت‌ـپوشش نیز نتایج مشابهی نشان دهند. در ادامه به رسم آن‌ها می‌پردازیم:

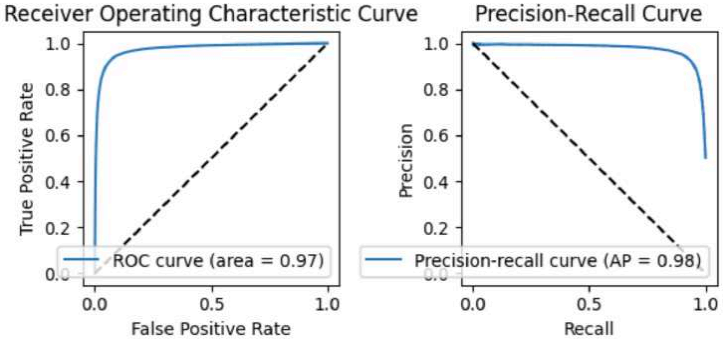



TF-IDF (آ)

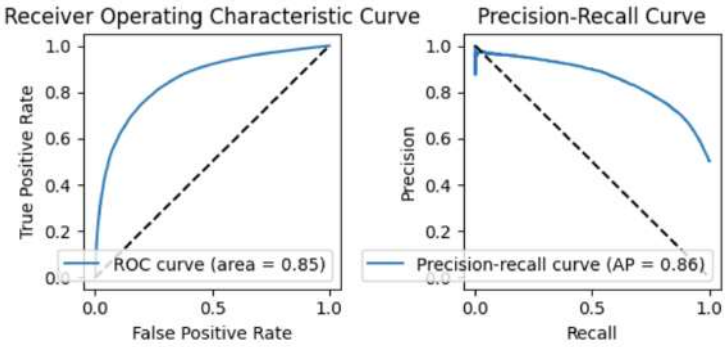



TF-IDF - ۲ (ب)

شکل ۱۴.۳: نمودارهای مشخصه عملکرد و صحت پوشش جنگل تصادفی




CountVec (آ)




Vader (ب)

شکل ۱۵.۳: نمودارهای مشخصه عملکرد و صحت پوشش جنگل تصادفی

همانطور که پیش‌بینی می‌شد، سه نمودار اول به نقاط بهینه نزدیک‌تر بوده و نتایج بهتری را نشان می‌دهند.
لازم به ذکر است که در الگوریتم قبل مساحت زیر نمودار ROC برابر با ۰/۷۴ و مساحت زیر نمودار Precision-Recall برابر با ۰/۷۱ بود (شکل ۱۰.۳ ) ،ولی در این الگوریتم این مساحت‌ها به ترتیب برابر با ۰/۸۵ و ۰/۸۶ هستند (شکل ۱۵.۳ ) که این نشان می‌دهد روش بردار سازی Vader در این الگوریتم نسبت به الگوریتم قبل نتایج بهتری داشته است.

## ۴.۳ دسته بند بیز ساده

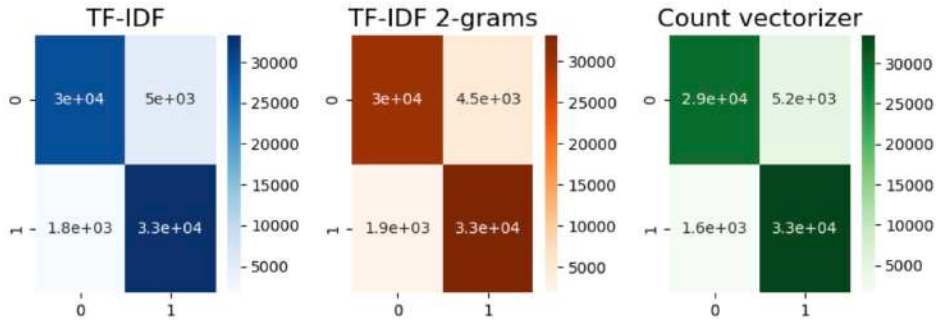
پارامترهایی که در این مدل تلاش برای بهبود آن‌ها می‌کنیم عبارتند از:

- alpha: در الگوریتم Naive Bayes اگر یکی از ویژگی‌ها در مجموعه داده آموزشی وجود نداشته باشد، احتمال محاسبه شده برای آن توسط الگوریتم صفر می‌شود. برای جلوگیری از این اتفاق، از پارامتر آلفا استفاده می‌شود. در واقع با این کار پس از محاسبه هر احتمال، مقدار مثبتی به آن اضافه می‌کنیم که این مقدار را اندازه آلفا تعیین می‌کند.
- fit prior:دامنه این پارامتر مقادیر صفر و یک است. اگر مقدار آن را برابر با ۱ قرار دهیم، مدل احتمال هر کلاس را از داده آموزشی یاد می‌گیرد. در غیر این صورت، مدل در نظر می‌گیرد که احتمال وقوع کلاس‌ها با هم برابر است.

این الگوریتم را نمی‌توان روی داده‌ای که از روش Vader Sentiment Score بدست آمده آموزش داد زیرا در این نوع بردارسازی، ممکن است داده‌های ستون‌هایمان منفی باشد و این موضوع با فرض دسته‌بند بیز مبنی بر اینکه ویژگی منفی نداریم، همخوانی ندارد.
بهترین پارامترهای بدست آمده عبارتند از:

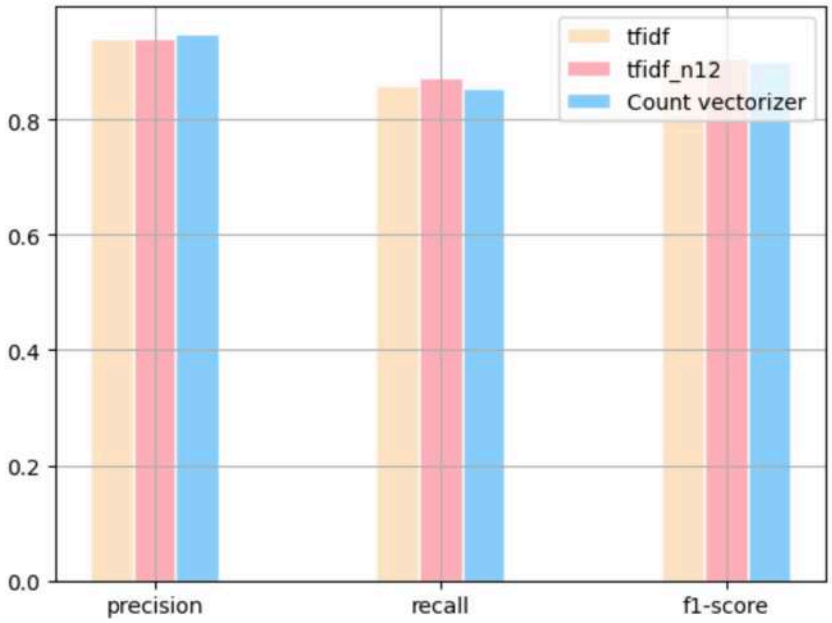
| Method | alpha | fit prior |
|---|---|---|
| TF-IDF | 0.1 | False |
| TF-IDF(n=(1,2)) | 1.0 | False |
| CountVectorizer | 0.1 | False |

حال با استفاده از پارامترهای بهینه بدست آمده، مدل را آموزش می‌دهیم. نتایج در شکل ۱۶.۳ نشان داده شده است.



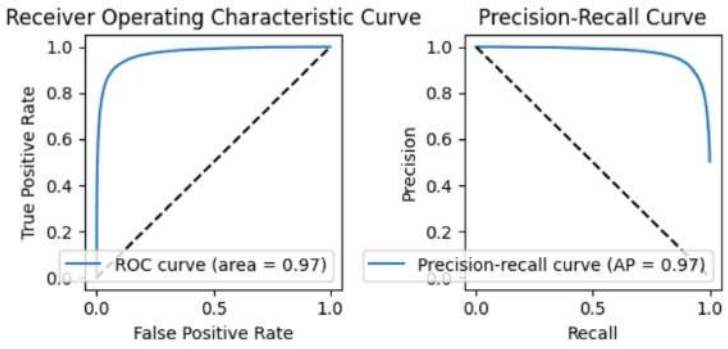

شکل ۱۶.۳: بردار درهم ریختگی الگوریتم طبقه‌بند بیز

| metric | TF-IDF | TF-IDF 2 | CountVec |
|---|---|---|---|
| precision | 0.942 | 0.939 | 0.949 |
| recall | 0.858 | 0.873 | 0.854 |
| F1-score | 0.898 | 0.905 | 0.899 |
| accuracy | 0.902 | 0.908 | 0.904 |

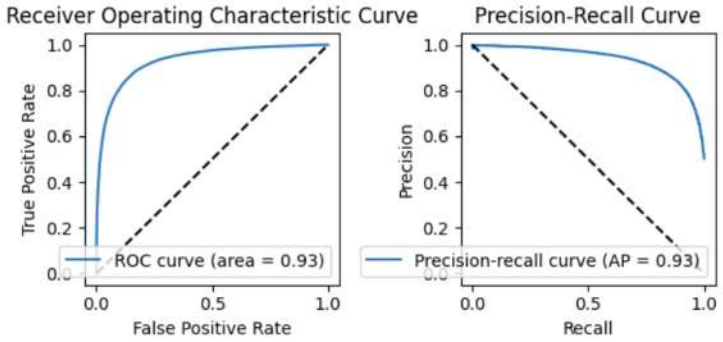



شکل ۱۷.۳: نمودار نتایج دسته‌بند بیز

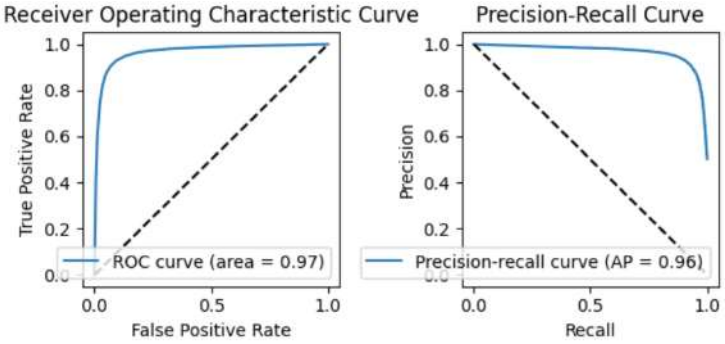
همانطور که مشاهده می‌کنید، هر سه روش برداری متن توانستند نتایج خوبی به دست آورند. نتایج نشان می‌دهد که معیارهای Accuracy و F1-score برای TF-IDF-2 و Count Vec. بالاتر از TF-IDF هستند، در حالی که معیارهای Recall و Precision برای TF-IDF و Count Vec. بالاتر از TF-IDF-2 هستند. بنابراین، برای این مجموعه داده، به نظر می‌رسد که Count Vec. روش برداری مناسب‌تری برای الگوریتم باشد.




TF-IDF (آ)




TF-IDF - ۲ (ب)




CountVec (ج)

شکل ۱۸.۳: نمودارهای مشخصه عملکرد و صحت پوشش دسته‌بند بیز

با توجه به نمودارهای کشیده شده و مساحت زیر آن‌ها نیز به این نتیجه می‌رسیم که داده مدل Count Vectorization بهترین نتیجه را داده است.

## ۵.۳ شبکه‌های عصبی

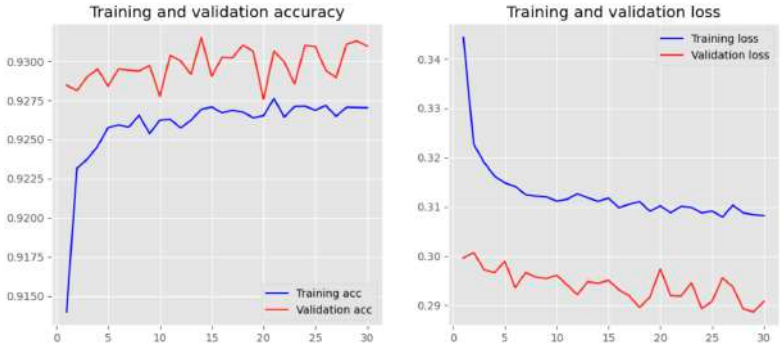
### ۱.۵.۳ شبکه عصبی ساده

ساختار شبکه عصبی آموزش داده شده در شکل ۱۹.۳ قابل مشاهده است. شبکه عصبی فوق را به تعداد ۳۰ دور آموزش می‌دهیم. سپس دقت آن را روی داده‌های تست حساب می‌کنیم.دقت حاصل برابر با ۰٫۹۳۱۰ بدست می‌آید. نتایج آموزش شبکه عصبی در شکل ۲۰.۳ نمایش داده شده است.

| Layer (type) | Output Shape | Param # |
|---|---|---|
| dense_4 (Dense) | (None, 64) | 631616 |
| dropout_3 (Dropout) | (None, 64) | 0 |
| dense_5 (Dense) | (None, 32) | 2080 |
| dropout_4 (Dropout) | (None, 32) | 0 |
| dense_6 (Dense) | (None, 16) | 528 |
| dropout_5 (Dropout) | (None, 16) | 0 |
| dense_7 (Dense) | (None, 1) | 17 |

Total params: 634241 (2.42 MB)
Trainable params: 634241 (2.42 MB)
Non-trainable params: 0 (0.00 Byte)

شکل ۱۹.۳: ساختار شبکه عصبی




شکل ۲۰.۳: آموزش شبکه عصبی

### ۲.۵.۳ شبکه‌های از پیش آموزش داده شده

در این بخش از شبکه برت استفاده شده. در استفاده از این نوع شبکه، نیاز به بردار سازی داده‌های ورودی نیست. ابتدا متن ورودی را توکن‌بندی کرده و به فرمت ورودی BERT تبدیل می‌کنیم. مدل با استفاده از بهینه‌ساز Adam بر روی مجموعه‌داده آموزشی دارای برچسب، آموزش می‌بیند و وزن‌های قبلی آن به‌روز رسانی می‌شوند. مدل را ۳ بار روی داده آموزشی آموزش می‌دهیم و پس از بهبود وزن‌ها، مدل بر روی یک مجموعه آزمون ارزیابی می‌شود و دقت آن حساب می‌شود. دقت داده آزمایشی پس از طی کردن این مراحل برابر با ۰/۹۷۴۴ می‌شود که از نتایج تمامی مدل‌های یادگیری ماشین بهتر است. هرچند که برای ۳ درصد پیشرفت نسبت به بهترین مدل قبلی، حدود ۳۰ ساعت تلاش به بهبود وزن‌های این مدل کردیم.

# فصل ۴

# جمع بندی

در این پایان نامه، با بررسی ادبیات و تحلیل داده‌های موجود در پلتفرم‌های آنلاین، مدلی با دقت بالا برای شناسایی افراد در معرض خطر خودکشی پیشنهاد دادیم. تحلیل داده‌های متنی از جمله متون نوشته شده توسط افراد در شبکه‌های اجتماعی، ایمیل‌های ارسال شده و پیام‌های متنی ارسال شده به دیگران، به عنوان یکی از روش‌های مؤثر در شناسایی افراد در معرض خطر خودکشی، مورد استفاده قرار گرفت. با استفاده از متدهای یادگیری ماشین و شبکه‌های عصبی، الگوهای مشترک در متون مورد بررسی شناسایی شده و مدلی برای شناسایی افراد در معرض خطر توسعه داده شده است.

نتایج حاصل از پایان نامه نشان می‌دهد که استفاده از تحلیل داده‌های متنی به عنوان یک روش نوین در شناسایی افراد در معرض خطر خودکشی، قابلیت‌های بسیاری را برای بهبود خدمات بهداشتی و درمانی فراهم می‌کند. با استفاده از مدل‌های پیشنهاد شده در این پایان نامه، می‌توان به صورت زودهنگام افرادی را شناسایی کرد که در معرض خطر خودکشی هستند و به آن‌ها کمک کرد تا به موقع دریافت خدمات ارزیابی و درمانی شوند.

نتایج آزمایش مدل‌ها نشان داد که بیشترین دقت با استفاده از مدل BERT به عنوان یک شبکه‌ی عصبی پیش‌آموزش دیده و فاین‌تیون شده، ۹۷ درصد به دست آمده است. همچنین، مدل‌های یادگیری ماشین ذکر شده که از روش‌های TF-IDF و CountVectorization برای بردارسازی استفاده کردند، دارای دقت‌هایی بین ۸۶ تا ۹۳ درصد بودند. بالاترین دقت نیز از مدل رگرسیون لوجستیک بدست آمد و نتایج این مدل تقریبا برابر با شبکه عصبی استفاده شده در این پروژه است.

# واژه‌نامه

داده کاوی متن ..... Text Mining
یادگیری ماشین ..... Machine Learning
شبکه‌های عصبی ..... Neural Networks
هوش مصنوعی ..... Artificial Intelligence
الگوریتم‌های بیزی ..... Bayesian Algorithms
الگوریتم‌های خوشه‌بندی ..... Clustering Algorithms
شبکه‌های عصبی عمیق ..... Deep Neural Networks
ماشین بردار پشتیبان ..... Support Vector Machine
شبکه‌های مولد ..... Generative Adversarial Networks
یادگیری نظارت شده ..... Supervised Learning
تابع هدف ..... Objective Function
کاهش گرادیان ..... Gradient Descent
یادگیری نظارت نشده ..... Unsupervised Learning
یادگیری نیمه نظارتی ..... Semi-Supervised Learning
لایه‌های نیمه نظارتی ..... Semi-Supervised Layers
یادگیری تقویتی ..... Reinforcement Learning
الگوریتم‌های طبقه‌بندی ..... Classification Algorithms
رگرسیون ..... Regression
تابع میانگین مربعات خطا ..... MSE
تابع سیگموید ..... Sigmoid Function
کمینه‌های موضعی ..... Local Minimums
کمینه‌های کلی ..... Global Minimums
ابر صفحه ..... Hyperplane
فاصله حاشیه ..... Margin Distance
جنگل تصادفی ..... Random Forest Algorithm
بیش‌برازش ..... Overfitting
دسته‌بند بیز ..... Naive Bayes Classifier

شرط ..... Evidence
فرضیه ..... Hypothesis
ساده ..... Naive
شبکه‌های عصبی مصنوعی ..... Artificial Neural Network
پس‌انتشار خطا ..... Backpropagation
آکسون‌ها ..... Axon
دندریت‌ها ..... Dendrite
سیناپس‌ها ..... Synapse
لایه ورودی ..... Input Layer
لایه مخفی ..... Hidden Layer
لایه خروجی ..... Output Layer
وزن‌ها ..... Weights
تابع فعال‌سازی ..... Activation Function
الگوریتم‌های یادگیری ..... Learning Algorithms
اریبی‌ها ..... Biases
شبکه‌های عصبی از پیش آموزش داده شده ..... Pre-trained Neural Networks
برت ..... BERT
پردازش زبان طبیعی ..... Natural Language Processing
تبدیل‌کننده ..... Transformer
کدگذار ..... Encoder
مساله ..... Task
مدل کردن زبانی ..... Language Modeling
پیش‌بینی عبارت بعدی ..... Next Sequence Prediction
تنظیم ..... Fine-tune
کم‌برازش ..... Underfitting
جستجوی خطی ..... Grid Search
جستجوی تصادفی ..... Random Search
دقت ..... Accuracy
فراخوانی ..... Recall
دقت پیش‌بینی ..... Precision
بازخوانی ..... F1-score
ریشه‌یابی کلمات ..... Lemmatization
تبدیل کلمات به بردار ..... Vectorization
مبتنی بر احساسات ..... Sentiment-Based
کلمات منفی یا مثبت کننده ..... Negation Reversal
بر زدن ..... Shuffle
ماتریس درهم‌ریختگی ..... Confusion Matrix

نمودار مشخصه عملکرد ..... Receiver Operating Characteristic plot
منحنی صحت-پوشش ..... Precision-Recall Curve
نرخ مثبت صحیح ..... True Positive Rate
نرخ مثبت کاذب ..... False Positive Rate
منظم سازی ..... Regularization
هموار ساز ..... Smoothing

# کتاب‌نامه


[1] them Alpaydin (2020). Introduction to Machine Learning (Fourth ed.).

[2] amuel, Arthur (1959). "Some Studies in Machine Learning Using the Game of Checkers". IBM Journal of Research and Development

[3] ussell, Stuart J.; Norvig, Peter (2010). Artificial Intelligence: A Modern Approach (Third ed.).

[4] ohri, Mehryar; Rostamizadeh, Afshin; Talwalkar, Ameet (2012). Foundations of Machine Learning.

[5] itchell, T. (1997). Machine Learning. McGraw Hill.

[6] lpaydin, Ethem (2010). Introduction to Machine Learning. MIT Press.

[7] hang, Bosen; Huang, Haiyan; Tibbs-Cortes, Laura E.; Vanous, Adam; Zhang, Zhiwu; Sanguinet, Karen; Garland-Campbell, Kimberly A.; Yu, Jianming; Li, Xianran (2023). "Streamline unsupervised machine learning to survey and graph indel-based haplotypes from pan-genomes"

[8] an Otterlo, M.; Wiering, M. (2012). Reinforcement learning and Markov decision processes. Reinforcement Learning. Adaptation, Learning, and Optimization

[9] avid A. Freedman (2009). Statistical Models: Theory and Practice. Cambridge University Press. p.26. "A simple regression equation has on the right hand side an intercept and an explanatory variable with a slope coefficient. A multiple regression e right hand side, each with its own slope coefficient"

[10] encher, Alvin C.; Christensen, William F. (2012), "Chapter 10, Multivariate regression – Section 10.1, Introduction", Methods of Multivariate Analysis, Wiley Series in Probability and Statistics, vol.709 (3rd ed.), John Wiley  Sons

[11] arzilai, Jonathan; Borwein, Jonathan M. (1988). "Two-Point Step Size Gradient Methods". IMA Journal of Numerical Analysis. 8 (1): 141–148. doi:10.1093/imanum/8.1.141.

[12] letcher, R. (2005). "On the Barzilai–Borwein Method". In Qi, L.; Teo, K.; Yang, X. (eds.). Optimization and Control with Applications. Applied Optimization. Vol. 96. Boston: Springer. pp.235–256. ISBN 0-387-24254-6.

[13] osmer, David W.; Lemeshow, Stanley (2000). Applied Logistic Regression (2nd ed.). Wiley. ISBN 978-0-471-35632-5

[14] ortes, Corinna; Vapnik, Vladimir N. (1995). "Support-vector networks". Machine Learning. 20 (3): 273–297. doi:10.1007/BF00994018.

[15] garap, A. F. M. (2018, February). A neural network architecture combining gated recurrent unit (GRU) and support vector machine (SVM) for intrusion detection in network traffic data. In Proceedings of the 2018 10th international conference on machine learning and computing (pp. 26-30)

[16] en-Hur, Asa; Horn, David; Siegelmann, Hava; Vapnik, Vladimir N. ""Support vector clustering" (2001);". Journal of Machine Learning Research. 2: 125–137

[17] o, Tin Kam (1995). Random Decision Forests (PDF). Proceedings of the 3rd International Conference on Document Analysis and Recognition, Montreal, QC, 14–16 August 1995. pp. 278–282

[18] oyce, James (2003), "Bayes' Theorem", in Zalta, Edward N. (ed.), The Stanford Encyclopedia of Philosophy (Spring 2019 ed.), Metaphysics Research Lab, Stanford University, retrieved 2020-01-17

[19] . Krose and V. D P. Smagt, An Introduction to NeuralNetworks, University of Amsterdam, Amsterdam, Switzer-land, 1996.

[20] ussell, Ingrid. "Neural Networks Module". Archived from the original on May 29, 2014.

[21] evlin, Jacob; Chang, Ming-Wei; Lee, Kenton; Toutanova, Kristina (11 October 2018). "BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding"

[22] laesen, Marc; Bart De Moor (2015). "Hyperparameter Search in Machine Learning".

[23] hicco D (December 2017). "Ten quick tips for machine learning in computational biology".

[24] iu, H.; Christiansen, T.; Baumgartner, W. A.; Verspoor, K. (2012). "BioLemmatizer: A lemmatization tool for morphological processing of biomedical text". Journal of Biomedical Semantics.

[25] ajaraman, A.; Ullman, J.D. (2011). "Data Mining" (PDF). Mining of Massive Datasets. pp. 1–17

[26] ong Anh Ho, Duong Huynh-Cong Nguyen, Danh Hoang Nguyen, Linh Thi-Van Pham, Duc-Vu Nguyen, Kiet Van Nguyen, Ngan Luu-Thuy Nguyen. "Emotion Recognition for Vietnamese Social Media Text". In Proceedings of the 2019 International Conference of the Pacific Association for Computational Linguistics (PACLING 2019), Hanoi, Vietnam (2019).

[27] owers, David M. W. (2011). "Evaluation: From Precision, Recall and F-Measure to ROC, Informedness, Markedness and Correlation". Journal of Machine Learning Technologies.

[28] awcett, Tom (2006). "An Introduction to ROC Analysis" (PDF). Pattern Recognition Letters.

**Abstract**

Suicide is a major public health concern, and early identification of individuals at risk of suicidal behavior can be a critical step in preventing suicide. However, identifying individuals at risk of suicide can be a challenging task, as suicidal behavior is often a complex and multifaceted phenomenon that can be difficult to detect. In recent years, text mining techniques and machine learning have been increasingly used to analyze text data and identify patterns that can be indicative of suicidal behavior.

The aim of this thesis is to develop a high-precision model that can identify texts that indicate suicidal behavior using text mining techniques and machine learning. To achieve this aim, text data from the social media platform Reddit was used, as it provides a rich source of text data that can be used to analyze the language of individuals who may be at risk of suicidal behavior. A combination of machine learning and deep learning techniques were used to develop the model, including Support Vector Machines (SVM), Random Forests, and Neural Networks.

The developed model achieved high accuracy in identifying texts that indicate suicidal behavior, demonstrating the potential of text mining and machine learning in identifying individuals at risk of suicide. This thesis can serve as a basis for further research in this field and provide solutions for improving public health by enabling early identification and intervention for individuals at risk of suicidal behavior.

College of Science
School of Mathematics, Statistics, and Computer Science

# Investigating the relationship between a person's written text and their suicide using text mining and machine learning methods

**Narges Baba Ahmadi**

Supervisor: Dr. Babaali

A thesis submitted in partial fulfillment of the requirements for the degree of B.Sc. in Computer Science

summer 2023