

بسم الله الرحمن الرحيم

پروژه پردازش متن

برای محصولات دیجیکالا

تیم ارائه دهنده: داده کاوان فونیکس

ناظر و سرپرست تیم: دکتر علیرضا وفایی صدر

درخواست دهنده: ستاد علوم شناختی IPM

۱۳۹۸

فهرست مطالب

۲	-----	مقدمه
۲	-----	تیم داده کاوان فونیکس
۲	-----	تعریف پروژه
۲	-----	آشنایی با دیتاست و داده ها
۳	-----	پیش پردازش و تمیز کردن داده
۴	-----	آنالیز و بصری سازی داده
۱۰	-----	کلاسه بندی کردن ستون هدف
۱۳	-----	ساخت بردار اعداد از متون
۱۴	-----	مدلسازی داده و یادگیری ماشین
۱۸	-----	بصری سازی مدل
۲۰	-----	کارها و تحقیقات آتی

مقدمه

پردازش متن یا (Text Mining) که به آن متن کاوی (Natural Language Processing) هم گفته می شود به مجموعه عملیاتی (Processes) بر روی متن اطلاق می شود که متن را برای کامپیوتر قابل فهم کند. زیرا کامپیوتر و ماشین عدد را می فهمد و نه متن و زبان طبیعی را. بعد از پردازش متن می توان از قدرت کامپیوتر در محاسبات و یادگیری ماشین استفاده کنیم تا اطلاعات ارزشمندی را از میان این داده ها استخراج کنیم. پردازش متن را می توان در زیرشاخه هی هوش مصنوعی و یا علم داده قرار داد.

تیم داده کاوی فونیکس

تیم ما، یعنی داده کاوی فونیکس در کارگاه آموزشی علم داده در پژوهشگاه فیزیک پژوهشگاه دانش های بنیادین IPM برای انجام پروژه های یادگیری ماشین و علم داده با تدریس و هدایت دکتر علیرضا وفایی صدر تشکیل شد. ما یک تیم شش نفره هستیم که بر روی پروژه هایی از جمله تشخیص و پیش بینی عیوب توربین های گازی شرکت نفت و گاز توربو تک و پروژه متن کاوی داده های توییتر کار کرده ایم و همچنان در حال کار بر روی پروژه های مختلف هستیم.

تعريف پروژه

هدف از انجام این پروژه، انجام متن کاوی بر روی نظرات مختلف در مورد محصولات فروشگاه اینترنتی دیجیکا است. تحلیل نظرات کاربران بر اساس تعداد لایک ها و دیسلایک های موجود، مزایا و معایبی که در در هر نظر برای محصول بیان شده است.

آشنایی با دیتاست و داده ها

فایل دیتاستی که به تیم برای پردازش و تحلیل داده شد، شامل ۱۰۰۰۰۰ داده (نظرات در مورد محصولات دیجیکالا) بود دارای ۱۲ ستون زیر:

شناسه محصول

عنوان محصول به فارسی

دسته بندی (عنوان) محصول به انگلیسی

شناسه کاربر

تعداد لایک های هر کامنٰت

تعداد دیسلایک های هر کامنٰت

وضعیت تایید شدن کامنٰت

کاربر محصول را توصیه می کند یا نه (شامل recommended, not_recommended, no_idea)

عنوان کامنٰت

کامنٰت محصول

مزایای محصول

معایب محصول

قرار شد بر اساس تعداد لایک ها و دیسلایک های هر نظر، کار تحلیل نظرات انجام شود و همچنین یک دسته بندی برای تعداد لایک ها و دیسلایک ها انجام شود به طوری که در آینده پیش بینی شود یک نظر مشخص چه تعداد لایک و دیسلایک خواهد داشت.

پیش پردازش و تمیز کردن داده

برای پردازش متون (چه فارسی و چه انگلیسی) اولین قدم پیش پردازش متن و تمیز کردن متون است که به آن Text Preprocessing گفته می شود. زیرا نظرات و متونی که غیر ادبی و غیر کتابی باشند و در شبکه های اجتماعی توسط مردم عام نوشته شده باشد بسیار در هم و دارای نویز بوده زیرا هر شخص بر اساس سلیقه خود اقدام به نگارش متن می کند. پس باید تمام متون به حالتی یکپارچه و استاندارد دربیايد و کلمات اضافی و بی استفاده از آنها حذف شود.

برای همین با استفاده از کتابخانه هضم Hazm کار تمیز کردن و نرمال سازی متن را انجام دادیم به صورت زیر:

نرمالسازی متون و حذف فاصله ها و علایم و اعراب گذاری ها.

حذف کلمات پر تکرار و بی استفاده StopWords ها.

حذف شکلک ها، حذف لینک ها، حذف Tab ها. حذف ایمیل، حذف کلمات انگلیسی، حذف کاراکترها و سمبل های خاص مثل علامت سوال، علامت تعجب، کروشه و ...

حذف فاصله های اضافی بین کلمات و حذف فاصله های اضافی بین سطرها.

انجام عملیات lemmatization یا برگرداندن کلمات به اصل ریشه. مثلاً تبدیل "رفتم" به "رو" و ...

این عملیات پاکسازی متون را برای ستون های متنی در دیتاست مثل ستون نظرات، ستون مزایا، ستون معایب و ستون عنوان محصول انجام دادیم.

همچنین در ۴ ستون فوق اگر دارای مقدار خالی بودند، با متن پیش فرض "فیلد خالی" پر کردیم.

برای ستون توصیه ها (recommend) هم عملیات mapping را بر اساس ارزش هر مقدار انجام دادیم. بدین صورت که داده های خالی را در این ستون همان no_idea یا نظری ندارم، در نظر گرفتیم و سپس بر اساس وزن هر مقدار، به مقدار "توصیه نمی کنم" عدد صفر، به مقدار "نظری ندارم" عدد ۱ و به مقدار "توصیه می کنم" عدد ۲ را اختصاص دادیم.

در نهایت تصمیم گرفتیم که حاوی لایک و دیسلایک همزمان صفر هستند را حذف کنیم چون اطلاعاتی را برای پیش بینی به ما نخواهند داد و ارزش چندانی برای ما ندارند.

کارهایی که می توان برای پیش پردازش متن انجام داد بسیارند و زمان زیادی خواهد برد و به علت کمبود وقت ما به همین روش ها بسته کردیم.

عملیات پیش پردازش متن در فایل Text Preprocessing.ipynb موجود است.

دیتاست پاکسازی شده را در فایلی به اسم digi_clean2.xlsx قرار دادیم. عملیات پیش پردازش داده و پاکسازی متون توسط آرمیتا رضوی انجام شد.

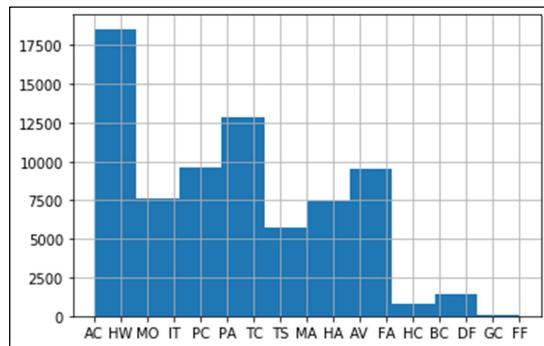
آنالیز داده و بصری سازی داده

برای اینکه در ک کاملی از داده ها داشته باشیم اقدام به آنالیز و تصویر سازی داده ها کردیم تا دید بهتری از داده ها داشته باشیم. همچنین کاربر با نگاه کردن به نمودار و شکل ها راحت تر متوجه داده می شود تا اینکه بخواهد کدهای آماری را بخواند.

آنالیز داده های پاکسازی شده انجام شده و همچنین نظرات و سطرهایی که دارای لایک و دیسلایک همزمان صفر بوده اند و برایمان ارزشی نداشتند را در این آنالیز و تصویر سازی، حذف کرده ایم. که از ۱۰۰ هزار سطر به مقدار ۷۳۴۶۹ سطر رسیدیم. ۲۶۵۳۱ کامنت، همزمان بدون لایک و دیسلایک بودند.

در این بخش گزارش به مهمترین آنالیزها می پردازیم:

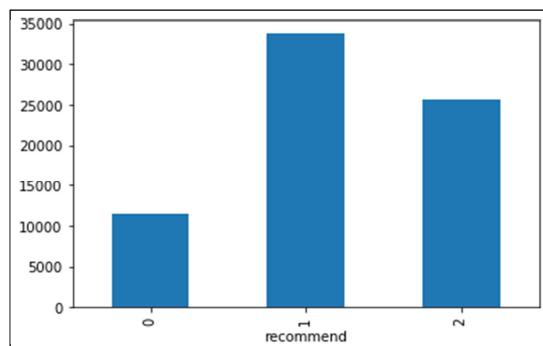
* بیشترین نظرات مربوط به محصولات AC و HW و کمترین مربوط به GC و FF است. (شکل ۱)



شکل ۱

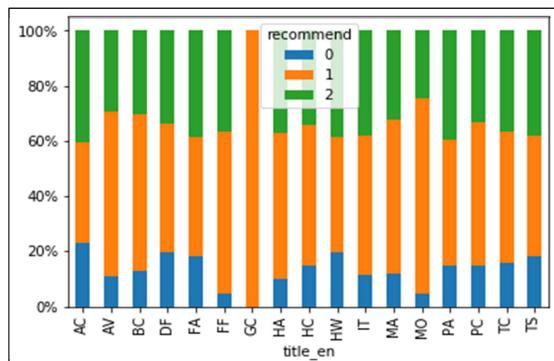
* از بین ۷۳۴۶۹ کامنت، حدود ۱۲۰۰۰ نظر حاوی کلید واژه‌ی "توصیه نمی کنم" است. حدود ۳۴۰۰۰ نظر حاوی کلید واژه‌ی "نظری ندارم" است و حدود ۲۵۰۰۰ نظر حاوی کلید واژه‌ی "توصیه می کنم" است. (شکل ۲)

0 : توصیه نمی کنم – 1 : نظری ندارم – 2 : توصیه می کنم



شکل ۲

مقادیر فیلد توصیه بر اساس هر دسته بندی محصولات را در شکل ۳ مشاهده می کنید:



شکل ۳

برای مثال در خرید محصولات AC حدود ۲۰٪ توصیه نشده است، ۴۰٪ نظری نداشته اند و ۴۰٪ هم توصیه کرده اند. جالب است که در دسته محصولات GC کلا هیچ توصیه ای نشده است و همگی گفته اند که نظری ندارند!

همچنین در این آنالیز، برای بصری سازی داده ها تعداد لایک ها و دیسلایک ها را به سه کلاس تقسیم کردیم:

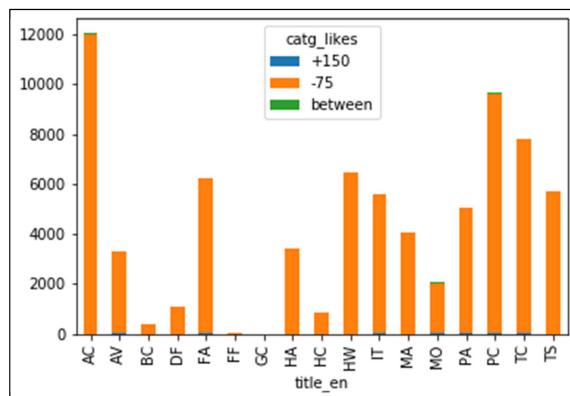
لایک ها و دیسلایک های زیر ۷۵ تا. لایک ها و دیسلایک های بین ۷۵ تا ۱۵۰ تا و لایک ها و دیسلایک های بالای ۱۵۰ تا که آماری بدین شکل به دست آمد:

تعداد کامنت های زیر ۷۵ تا لایک: ۷۳۳۴۳ عدد - تعداد کامنت های بین ۷۵ و ۱۵۰ لایک: ۱۰۲ عدد - تعداد کامنت های بالای ۱۵۰ تا لایک: ۲۴ عدد

تعداد کامنت های زیر ۷۵ تا دیسلایک: ۷۳۳۷۹ عدد - تعداد کامنت های بین ۷۵ و ۱۵۰ دیسلایک: ۵۹ عدد - تعداد کامنت های بالای ۱۵۰ تا دیسلایک: ۳۱ عدد

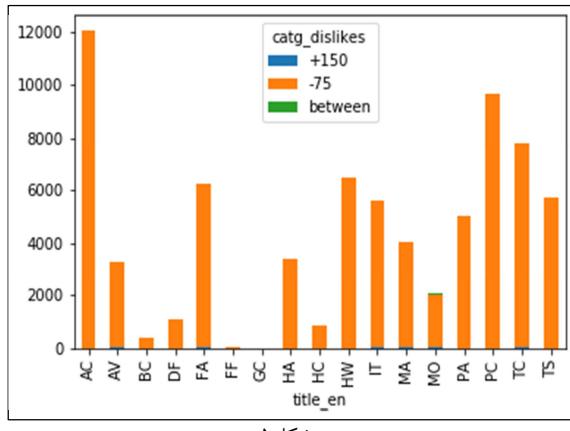
که تقریباً تفاوت فاحشی را بین لایک و دیسلایک نشان نمی دهد.

همچنین تعداد لایک ها و دیسلایک ها را برای هر دسته بندی محصول روی نمودار بردیم: (شکل ۴ و شکل ۵)



شکل ۴

همانطور که مشاهده می شود اکثر محصولات لایک های زیر ۷۵ تا دارند و محصولات AC بیشترین کامنت های حاوی لایک و محصولات دسته بندی GC, FF, BC کمترین کامنت های حاوی لایک ها را دارند و همگی زیر ۷۵ تا لایک هستند.



شکل ۵

در نمودار دیسلایک ها هم تفاوت چندانی با نمودار لایک ها دیده نمی شود که با توجه به نزدیک بودن تعداد لایک ها و دیسلایک ها امری طبیعی است. این آنالیزها توسط **آرمیتا رضوی** انجام شد.

کلمات ابری یا Word Cloud

سعی کردیم درکی از تعداد کلمات رایج و پر تکرار در کامنت ها و کلیدواژه های مزایا و معایب هر محصولات پیدا کنیم. برای همین اقدام به تولید word cloud برای کلمات پر تکرار در نظرات محصولات کردیم.

در ابتدا ۶۰ کلمه پر تکرار در کل نظرات و کامنت ها را تصویر سازی کردیم: (شکل ۶)



شكل ٦

در کل کامنت ها و از بین ۶۰ کلمه پر تکرار، کلمات "البته"، "جنس"، "گرفتم"، "هیچ" و "رنگ" بیشتر تکرار شده است.

سپس سراغ ۳۰ کلمات پر تکرار که در معاویب کالا ذکر شده است و بیشتر از ۱۵۰ دیسلایک خورده اند رفتیم و کلمات ابری را برای این نظرات رسم کردیم: (شکل ۷)



شکل ۷

در کل معاویب کالاهای از بین ۳۰ کلمه پر تکرار، کلمات "ندارد"، "کمی"، "عدم"، "ضعیف" و "پایین" بیشتر تکرار شده است که برای کلمات منفی در ستون معاویب کالا، طبیعی است.

سپس سراغ ۳۰ کلمات پر تکرار که در مزایای کالا ذکر شده است و بیشتر از ۱۵۰ لایک خورده اند رفتیم و کلمات ابری را برای این نظرات رسم کردیم: (شکل ۸)



در کل مزایای کالاهای از بین ۳۰ کلمه پر تکرار، کلمات "بوی"، "مناسب"، "شیک"، "سریع" و "قوت" بیشتر تکرار شده است که برای کلمات مثبت در ستون مزایای کالا، دور از انتظار نیست.

همچنین کلمات ابری و پر تکرار را برای کامنت ها بر اساس دسته بندی هر محصول و تعداد لایک ها و دیسلایک هایشان در آوردیم. و همچنین کلمات پر تکرار را برای نظراتی که در رابطه با چندین محصول خاص وجود داشته است هم در آنالیز آورده شده است که که توسط خانم ها **آرمیتا رضوی و زهراء نفریه** انجام شده که همه این نمودار ها و آنالیزها را می توانید در فایلهایی با اسمهای Text Exploration

مشاهده کنید. به علاوه، تمامی عکس های word cloud ها در پوشه word clouds تحویل داده خواهد شد.

تولید کلمات ابری بر اساس دسته بندی کالا توسط خانم زهرا نفریه:

کار بررسی کلمات پر تکرار در کامنت ها، مزایا و معایب کالا بر اساس هر دسته بندی کالا انجام شد که نتایج به صورت زیر است:

نمونه ای از دسته بندی لایک ها و دیسلایک ها که در ابتدای آنالیز داده ها بیان شد:

	طبقه بندی Like و Dislike	معیار طبقه بندی
1	High Like & Low dislike	Like = 150 dislike= -75
2	Low Like & Medium dislike	Like = -75 Dislike= between
3	Medium Like & Low dislike	Like= between Dislike = -75

دسته بندی محصولات:

[‘IT’ ‘AC’ ‘HW’ ‘MO’ ‘PC’ ‘PA’ ‘TC’ ‘TS’ ‘MA’ ‘HA’ ‘AV’ ‘FA’ ‘HC’ ‘BC’ ‘DF’ ‘GC’ ‘GF’ ‘FF’]

همچنین مجموع تعداد لایک ها و دیسلایک ها را برای کامنت ها بر طبق هر دسته بندی به دست آوردند:

دسته بندی محصولات	ستون نظرات یا کامنت ها			طبقه بندی لایک و دیسلایک
	مجموع لایک	مجموع دیسلایک		
1 IT	56	56		Low Like & Low dislike
2 AC	12	12		Low Like & Low dislike
3 HW	0	0		No Like & No dislike
4 MO	102	102		Medium Like & Medium dislike
5 PC	39	39		Low Like & Low dislike
6 PA	10	10		Low Like & Low dislike
7 TC	0	0		No Like & No dislike
8 TS	0	0		No Like & No dislike
9 MA	19	19		Low Like & Low dislike
10 HA	0	0		No Like & No dislike
11 AV	46	46		Low Like & Low dislike
12 FA	29	29		Low Like & Low dislike
13 HC	0	0		No Like & No dislike
14 BC	0	0		No Like & No dislike
15 DF	0	0		No Like & No dislike
16 GC	0	0		No Like & No dislike
17 GF	0	0		No Like & No dislike
18 FF	0	0		No Like & No dislike

کلاسه بندی کردن ستون هدف

هدف اصلی این پژوهه تعیین میزان لایک ها و دیسلایک ها در هر کامنت و پیش بینی بر اساس مدل های یادگیری ماشینی است که در آینده با توجه به هر کامنت بتوان میزان لایک ها و دیسلایک های آن را تشخیص داد.

از آنجا که مقادیر لایک ها و دیسلایک ها مقادیری پیوسته بودند تصمیم گرفتیم آنها را به مقادیر گسسته دسته بندی و کلاسه بندی کنیم. چهار کلاس را در نظر گرفتیم: کلاس لایک ها، کلاس خیلی لایک ها، کلاس دیسلایک ها و کلاس خیلی دیسلایک ها.

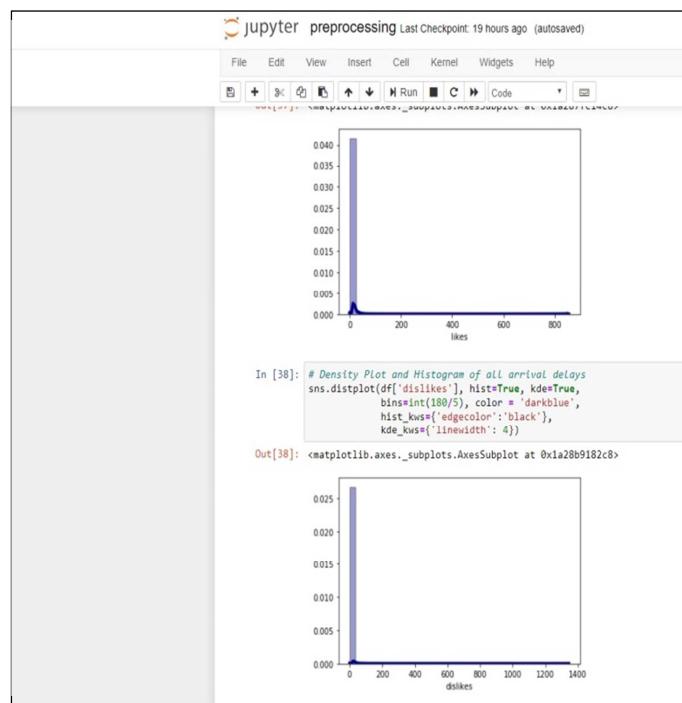
در ابتدا یک دید آماری از لایک ها و دیسلایک ها به دست آوردیم که در زیر مشاهده می کنید: (شکل ۹)

```
In [32]: df.likes.describe()
Out[32]: count    100000.000000
          mean     3.114790
          std      8.050031
          min      0.000000
          25%     0.000000
          50%     1.000000
          75%     4.000000
          max     854.000000
          Name: likes, dtype: float64

In [33]: df.dislikes.describe()
Out[33]: count    100000.000000
          mean     1.649460
          std      8.912705
          min      0.000000
          25%     0.000000
          50%     0.000000
          75%     1.000000
          max     1344.000000
          Name: dislikes, dtype: float64
```

شکل ۹

همینطور نمودار توزیع داده را برای این دو ستون رسم کردیم: (شکل ۱۰)



شکل ۱۰

متاسفانه توزیع داده های لایک و دیسلایک اصلا توزیع نرمالی نبود و برای کلاسه بندی کردن داده ها طوری که تعداد داده های یکسان در هر کلاس قرار گیرد و توازن بین کلاس ها برقرار شود مجبور شدیم از روش هایی غیری از مد و میانه و چارک ها استفاده کنیم.

روش اول کلاسه بندی ستون هدف:

یک روش که توسط آقای **وحید غفوریان** پیشنهاد شد بدین صورت بود که کلاس بندی لایک ها و دیسلایک ها را بر اساس عدد ۴ در نظر بگیریم به صورت زیر که مشاهده می کنید: (شکل ۱۱)

Like=0 and dislike=0	0
Like<4 and dislike<4	1
Like>=4 and dislike<4	2
Like<4 and dislike>=4	3
Like>=4 and dislike>=4	4

شکل ۱۱

در این نوع کلاس بندی بر اساس تعداد لایک ها و دیسلایک های کمتر و بیشتر از ۴، پنج نوع کلاس برای ستون هدف تعیین شد که البته لایک ها و دیسلایک های همزمان صفر هم به عنوان کلاس صفر، در نظر گرفته شده است.

از آنجا که تعداد داده ها در ۵ کلاس، متوازن نبود، از عملیات تکرار داده ها یا up-sampling استفاده شد و تعداد داده ها در هر کلاس به ۲۶۰۰۰ داده رسید تا توافق بین داده های کلاس ستون هدف برقرار شود. همین نوع کلاس بندی هم دوباره انجام شد ولی بدون کلاس صفر و کلاسی که لایک و دیسلایک همزمان صفر داشت را در نظر نگرفتیم و ستون هدف را ۴ کلاسه کردیم.

روش دوم کلاسه بندی ستون هدف:

در روش دیگری که خانم رقیه فرجی ارائه و پیشنهاد دادند بدین صورت لایک ها و دیسلایک ها را طبقه بندی کردیم که در ابتدا داده های پرت و یا outlier که خارج از بازه سه انحراف معیار STD بودند مشخص گردید و کنار گذاشته شد. با انجام اینکار آستانه جداسازی را میانگین در نظر گرفتیم و در نتیجه توافق و تقارن داده ها حول میانگین بهتر شد و کلاس بندی لایک ها و دیسلایک ها را روی مرزبندی جدید انجام دادیم. کلاس حاوی تعداد لایک هایی کمتر از میانگین، کلاس بزرگتر مساوی میانگین لایک ها و به همین ترتیب دو کلاس برای دیسلایک ها. جمعاً چهار کلاس برای ستون هدف به دست آمد. از داده های پرت بعداً برای داده های تست و آموزشی در مدل، استفاده کردیم.

برای اینکه تعداد داده های هر کلاس برابر شود از re-sampling استفاده شد و تعداد سطرها در هر کلاس به ۳۰۰۰ رسید. همچنین داده هایی که تعداد لایک ها و دیسلایک هایشان همزمان صفر بود را حذف کردیم و در کلاس بندی نیاوردیم.

روش سوم کلاسه بندی ستون هدف

در کلاسه بندی دیگری که توسط خانم فرجی پیشنهاد شد این بود که داده هایی که لایک های آنها صفر ولی دیسلایک آنها بیشتر از صفر هستند را به عنوان just-dislike درنظر بگیریم و بر عکس داده هایی که دیسلایک های آنها صفر ولی لایک آنها بیشتر از صفر هستند را به عنوان just-like درنظر بگیریم. با این روش، تعداد کلاس های ستون هدف شش کلاسه شد.

ساخت پرداز اعداد از متون

بعد از اینکه کار کلاسه بندی ستون های لایک و دیسلایک انجام شد تصمیم گرفتیم داده های را به الگوریتم های یادگیری ماشین و همچنین به شبکه عصبی بدهیم تا بتوان در آینده پیش بینی کرد که هر کامنت محصول در دیجیکالا در چه کلاسی از لایک و دیسلایک قرار می گیرد.

از آنجا که تمامی الگوریتم ها و مدل های یادگیری ماشین، ورودی را به صورت اعداد می گیرند و درکی از متن ندارند باید متن را به بردارهایی از اعداد تبدیل کرده و به هر کلمه وزن و ارزشی را اختصاص دهیم که به اینکار وکتورایز vectorize کردن متن گفته می شود.

روش های مختلفی برای اینکار وجود دارد که ما از دو روش TF-IDF برای کار با مدل Naive Bayes و همچنین از روش Tokenizer برای آماده سازی داده ها برای کار با شبکه های عصبی در مدل های deep learning استفاده کردیم.

اولتاً داده ها را به داده های train و validation تقسیم کردیم به نسبت ۷۰ به ۳۰.

ساخت پرداز متنی پایی ستون کامنت ها با TF-IDF : (شکل ۱۲)

```
[ ] 1 from sklearn.feature_extraction.text import TfidfVectorizer
[ ] 2 from sklearn.model_selection import train_test_split

[ ] 1 #Create the vectorizer
[ ] 2 vectorizer = TfidfVectorizer()

[ ] 1 X = df_cin['comment']
[ ] 2 y = df_cin['class']

[ ] 1 print(X.shape, y.shape)
[ ] (120716,) (120716,)

[ ] 1 #Tokenize and build vocab
[ ] 2 vectorizer.fit(X)

[ ] 1 # Split train data into train and validation
[ ] 2 train_x, valid_x, train_y, valid_y = train_test_split(X, y, test_size=0.25)

[ ] 1 print(train_x.shape,valid_x.shape)
[ ] (55101,) (18368,)

[ ] 1 # Use TF-IDF to Vectorize the train and validation data
[ ] 2 xtrain_tfidf = vectorizer.transform(train_x)
[ ] 3 xvalid_tfidf = vectorizer.transform(valid_x)

[ ] 1 # See train vectorized data
[ ] 2 xtrain_tfidf.data[:5]

[ ] array([0.09020482, 0.17425183, 0.12502186, 0.0779702 , 0.2279264 ])

[ ] 1 # See train vectorized data
[ ] 2 xvalid_tfidf.data

[ ] array([0.36161907, 0.14406568, 0.19714958, ..., 0.54033258, 0.60784316,
       0.58186544])

[ ] 1 # Word Dictionary
[ ] 2 print(vectorizer.vocabulary_)

[ ] 1 # Word Dictionary
[ ] 2 print(vectorizer.vocabulary_)
```

شکل ۱۲

ساخت بردار متن برای ستون کامنت ها با Tokenizer : (شکل ۱۳)

```
[ ] 1 from keras.preprocessing.sequence import pad_sequences
[ ] 2 from keras.preprocessing.text import Tokenizer
[ ]
[ ] 4 tokenizer = Tokenizer(num_words = vocab_size)
[ ] 5 tokenizer.fit_on_texts(train_x)
[ ] 6 word_index = tokenizer.word_index
[ ]
[ ] 8 train_sequences = tokenizer.texts_to_sequences(train_x)
[ ] 9 valid_sequences = tokenizer.texts_to_sequences(valid_x)

[ ] 1 # get only the top frequent words on train
[ ] 2 train_data = pad_sequences(train_sequences, padding = "post", maxlen = max_length)
[ ] 3 # get only the top frequent words on test
[ ] 4 valid_data = pad_sequences(valid_sequences, padding = "post", maxlen = max_length)
```

شکل ۱۳

مدلسازی داده و یادگیری ماشین

برای کار مدلسازی، از دو مدل استفاده کردیم، مدل MultinomialNB که همان مدلی از Bayes است از کتابخانه sk-learn در پایتون، و دیگری شبکه های عصبی کانولوشنی یا CNN از deep learning مدلهای

مدلسازی انجام شده توسط خانم رقیه فرجی

در ابتدا دیتاستی که دارای شش کلاس متوازن برای ستون هدف بود را به مدل MultinomialNB دادیم که دقیقی معادل ۴۲٪ داد. سپس دیتاست چهار کلاسه نامتوازن را که در فایل unbalance_classified.csv قرار دارد به مدل MultinomialNB داده شد که دقیقی معادل ۴۰٪ گرفتیم. مدلی از MultinomialNB در زیر دیده می شود: (شکل ۱۴)

```
Apply Naive Bayes

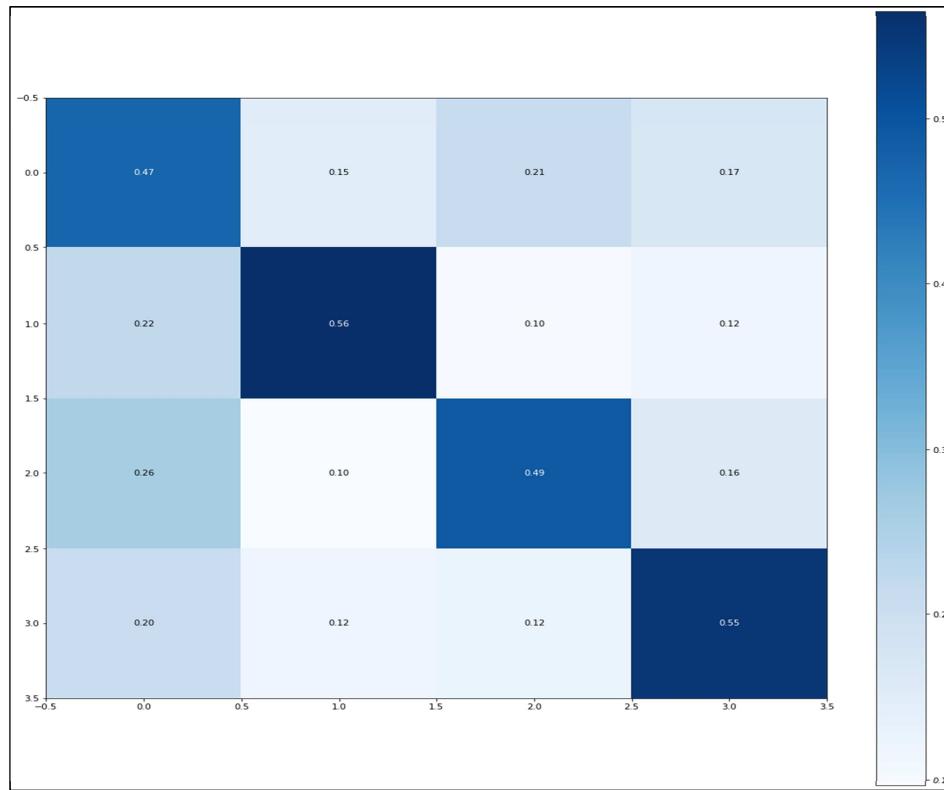
[ ] 1 # Naive Bayes training
[ ] 2 from sklearn.naive_bayes import MultinomialNB
[ ]
[ ] 4 # accuracy = train_model(MultinomialNB(alpha=0.2), xtrain_tfidf, train_y, xvalid_tfidf)
[ ] 5 # print ("Accuracy: ", accuracy)
[ ] 6 # pred=

[ ] 1 from sklearn.metrics import accuracy_score
[ ]
[ ] 3 # def train_model(classifier, feature_vector_train, label, feature_vector_valid, is_neural_net=False):
[ ] 4 # fit the training dataset on the classifier
[ ] 5 classifier=MultinomialNB(alpha=0.2)
[ ] 6 classifier.fit(xtrain_tfidf, train_y)
[ ] 7 # predict the labels on validation dataset
[ ] 8 pred = classifier.predict(xvalid_tfidf)
[ ] 9 print(accuracy_score(pred,valid_y))
[ ] 10 # return accuracy_score(pred, valid_y)

D: 0.42465156794425085
```

شکل ۱۴

کانفیوژن ماتریکس مدل MultinomialNB برای چهار کلاسه متوازن: (شکل ۱۵)



شکل ۱۵

هچنین کانفیوژن ماتریس چهار کلاسه برای داده های نامتوازن هم در عکس های پروژه موجود است.

سپس خانم فرجی دیتاست چهار کلاسه متوازن خود را که در فایل `balance_classified.csv` قرار دارد

به مدل شبکه عصبی داد و شبکه را آموزش داد که دقیقی معادل ۷۵٪ برای چهار کلاسه به دست آورد.

معماری شبکه عصبی را در شکل زیر مشاهده می کنید: (شکل ۱۶)

```
model = Sequential()
model.add(Embedding(input_dim = 120716 , output_dim = 30 , input_length = 30))
model.add(Reshape((30, 30, 1), input_shape = (30,30 )))
model.add(Dropout(0.5))
    # call convolution method defined above
model.add(convolution())

model.add(Flatten())
model.add(Dense(100))
model.add(BatchNormalization())
model.add(Activation('relu'))
model.add(Dropout(0.5))
model.add(Dense(10))
model.add(Activation('relu'))
model.add(BatchNormalization())
model.add(Dropout(0.5))
model.add(Dense(4))
model.add(Activation('sigmoid'))

adam = optimizers.Adam(lr = 0.001)

model.compile(loss='binary_crossentropy' , optimizer=adam , metrics=['accuracy'])
```

```

filter_sizes = [3, 4, 5]
def convolution():
    inn = Input(shape = (30, 30, 1))
    convolutions = []
    # we conduct three convolutions & poolings then concatenate them.
    for fs in filter_sizes:
        conv = Conv2D(filters = 100, kernel_size = (fs, 30), strides = 1, padding = "valid")(inn)
        nonlinearity = Activation('relu')(conv)
        maxpool = MaxPooling2D(pool_size = (30 - fs + 1, 1), padding = "valid")(nonlinearity)
        convolutions.append(maxpool)

    outt = concatenate(convolutions)
    model = Model(inputs = inn, outputs = outt)

    return model

```

شکل ۱۶

خلاصه مدلسازی خانم فرجی در جدول زیر دیده می شود: (شکل ۱۷)

تعداد کلاس بندی	متوازن یا نامتوازن	مدل	دقت پیش بینی مدل
۴	متوازن	Naive Bayes	٪۵۰
۴	نامتوازن	Naive Bayes	٪۴۳
۴	متوازن	CNN	٪۷۵
۴	نامتوازن	CNN	٪۵۰
۶	متوازن	Naive Bayes	٪۴۲

شکل ۱۷

تمامی مدل‌های این بخش در فایل final_model2.ipynb قرار دارد.

مدلسازی انجام شده توسط آقای وحید غفوریان

همانطور که گفتیم آقای غفوریان دو نوع کلاسه بندی متوازن برای ستون هدف تولید کرد. ۴ کلاسه و ۵ کلاسه. دیتاست چهار کلاسه در فایل digi_VahidEditionation_balanced_4class.csv و دیتاست پنج کلاسه در فایل digi_VahidEditionation_balanced_5class.csv قرار دارد.

هر دو دیتاست روی مدل شبکه عصبی دیگری آموزش دیده شد و برای ۵ کلاسه دقت ٪۶۲ و برای ۴ کلاسه دقت ٪۷۴ به دست آورد. (فایل final_model3.ipynb)

```

1 model = tf.keras.Sequential([
2     tf.keras.layers.Embedding(vocab_size, embedding_dim, input_length=max_length),
3     tf.keras.layers.Flatten(),
4     tf.keras.layers.Dense(6, activation='relu'),
5     tf.keras.layers.Dense(5, activation='softmax')
6 ])
7 model.compile(loss='categorical_crossentropy', optimizer='adam', metrics=['accuracy'])
8 model.summary()

```

Layer (type)	Output Shape	Param #
embedding_3 (Embedding)	(None, 128, 160)	1600000
flatten_3 (Flatten)	(None, 19200)	0
dense_6 (Dense)	(None, 6)	115206
dense_7 (Dense)	(None, 5)	35

Total params: 1,715,241
Trainable params: 1,715,241
Non-trainable params: 0

```

1 history = model.fit(train_data, train_labels, epochs = 20, validation_data = (valid_data, valid_labels), verbose = 1)
...

```

Train on 72800 samples, validate on 31200 samples

Epoch 1/20
72800/72800 [=====] - 58s 803us/sample - loss: 1.1744 - acc: 0.4376 - val_loss: 0.8547 - val_acc: 0.6146
Epoch 2/20
72800/72800 [=====] - 57s 784us/sample - loss: 0.6061 - acc: 0.7433 - val_loss: 0.6740 - val_acc: 0.7131

شکل ۱۸

مدلسازی انجام شده توسط آرمیتا رضوی

در ابتدا لایک های بزرگتر مساوی ۱ را یک کلاس در نظر گرفتیم و زیر ۱ را کلاسی دیگر.

سپس دیسلایک های بزرگتر مساوی ۲ را یک کلاس و زیر ۲ را کلاسی دیگر در نظر گرفتیم و ستون هدف را چهار کلاسه کردیم. ولی تعداد داده های کلاس را متوازن نکردیم و برای تست به مدل دادیم که برای مدل MultinomialNB دقต ۴۰٪ و برای شبکه عصبی دقت ۳۰٪ داد. امری طبیعی بود چون کلاس ها اصلاً متوازن نبودند و مرزبندی هم مرزبندی متقارنی نبود.

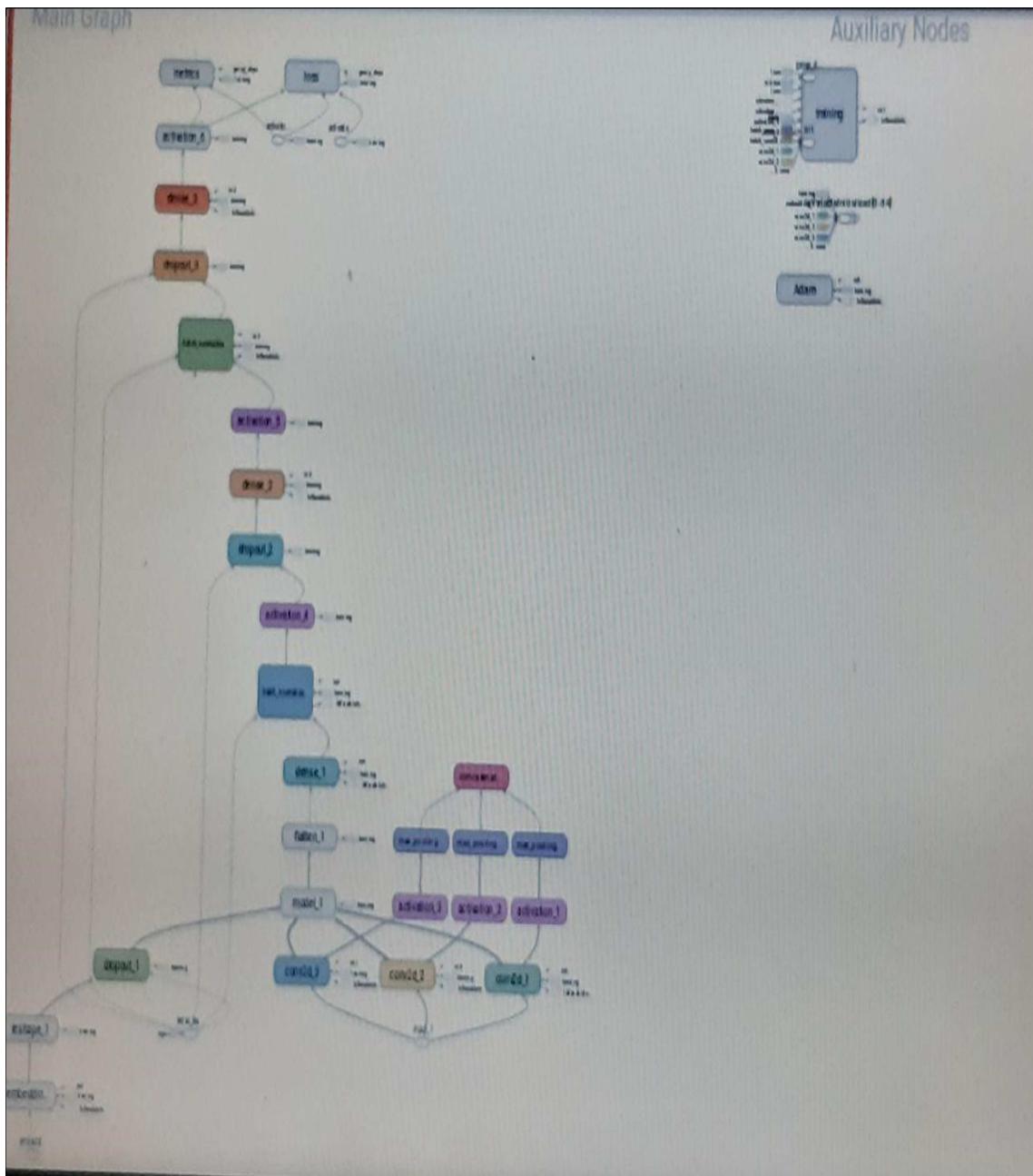
سپس دیتاست چهار کلاسه متوازن خانم فرجی (classified_3.csv) را بر روی مدل شبکه عصبی آقای غفوریان اجرا کرده و دقتی معادل ۶۹٪ گرفته شد.

نتیجه کلی مدلسازی به این صورت است که بهترین حالت، ۴ کلاسه کردن ستون هدف و به صورت داده های متوازن می باشد که بالاترین دقت ۷۵٪ را داده است.

فایل این مدلسازی به اسم final_model.ipynb است که مدل ۶۹٪ در آن قرار دارد.

بصري سازی مدل

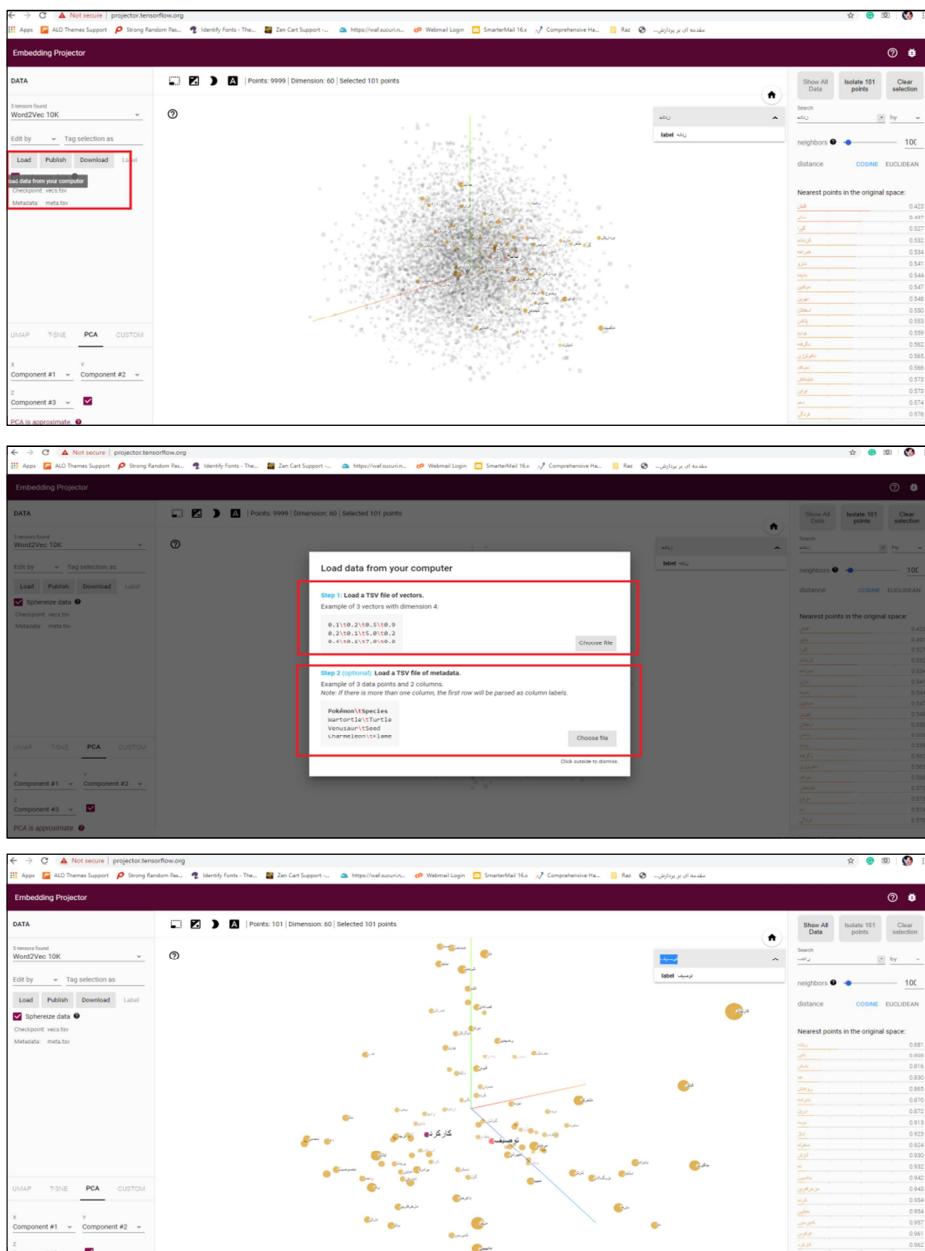
يك نمونه از معماري مدل شبکه عصبی که روی چهار کلاسه متوازن دقت ۷۵٪ داد را با tensor-board بصری سازی کردیم: (شکل ۱۹)



شکل ۱۹

متاسفانه وضوح عکس کیفیت خوبی ندارد ولی می توانید عکس مورد نیاز را در فolder images مشاهده کنید.

همچنین مدل شبکه عصبی ای که دقت ۶۹٪ داد را انتخاب کرده و وزنها و کلمات موجود در لایه‌ی Embedding آن را در دو فایل جدا به اسم‌های `meta.tsv` و `vecs.tsv` ذخیره کردیم و در سایت `projector.tensorflow.org` فایل‌ها را آپلود و نمایی سه بعدی از ماتریس بردار کلمات به دست آوردیم. در این بصری سازی می‌توان دید که کدام کلمات به هم نزدیکتر هستند و ارتباط بیشتری با هم دارند و همچنین می‌توان کلمات همسایه هر کلمه را پیدا کرد. این روش برای کلاستر بنده کلمات بسیار مفید است. در واقع این بصری سازی همان بصری سازی `word2vec` می‌باشد. این دو فایل در فایل‌های پروژه موجود است و می‌توانید با مراجعه به سایتی که ذکر شد و آپلود فایل‌ها تصویری کامل و جامع از ماتریس کلمات داشته باشید: (شکل ۲۰)



شکل ۲۰

کارها و تحقیقات آتی

از آنجا که تیم داده کاوان فونیکس فقط شش رو برای پردازش متن این دیتابست زمان داشت، ما تمام تلاش خود را کردیم که بهترین نتیجه را در این زمان کوتاه تحویل دهیم. دیدی کلی از داده به دست آوردیم، داده ها را پیش پردازش کرده و پاکسازی کردیم. تعداد کلمات موجود در هر کامنت را به دست آوردیم. یکسری آنالیز روی داده ها انجام داده و داده ها را تصویر سازی کرده و روی نمودار بردیم. کلمات پرتکرار را برای یکسری از کامنت ها بر اساس تعداد لایک ها و دیسلایک ها به دست آورده و word cloud آنها را کشیدیم. همچنین word cloud های مختلفی از کامنت ها و متون مزايا و معایب بر اساس دسته بندی کالاها و یکسری کالاهای خاص کشیدیم و کلمات مثبت و منفی را درآوردهیم که همگی در پوشه عکس ها موجود است.

به این نتیجه رسیدیم که بیشتر کاربران روی کامنت ها برای توصیه کردن خرید محصولات، نظری خاصی ندارند و بیشترین کامنت ها دارای لایک و دیسلایک زیر ۷۵ عدد هستند.

همچنین انواع مختلفی برای کلاسه بندی کردن میزان لایک ها و دیسلایک ها در نظر گرفتیم و برای هر نوع مرزبندی و کلاسه بندی، سعی کردیم داده ها را متوازن کنیم و روی هر کدام مدلسازی انجام دادیم. که در نهایت به دیتابست ۴ کلاسه و متوازن رسیدیم که دقیقی معادل ۷۵٪ روی شبکه عصبی به ما داد.

این فایل مستند به همراه تمامی کدها و عکس های پروژه در آدرس [github](https://github.com/phoenix-dataminers/Digikala) گروه فونیکس به آدرس زیر قرار دارد که می توانید با مراجعه به این آدرس، به فایل های پروژه دسترسی داشته باشید:

<https://github.com/phoenix-dataminers/Digikala>

کارهای بسیار زیادی می توان در آینده روی این دیتابست انجام داد. می توان عملیات پیش پردازش و feature engineering بهتر و بیشتری انجام داد. می توان ارتباط و شباهت بین هر کامنت را درآورد و همچنین کلمات کلیدی را از بین کامنت ها برای محصولات استخراج کرد.

می توان کار دسته بندی کامنت ها را یا همان کلاستریندی را انجام داد که بتوان کامنت ها را در سه دسته مثبت، منفی و خنثی برای sentiment analyze قرار داد و کارهای بسیاری که در صورت داشتن زمان و تمایل می شود انجام داد.

همچنین می توان با صرف وقت بیشتری و دادن کلمات مزايا و معایب کالا علاوه بر نظرات، دقت پیش بینی مدل را افزایش داد.

در آخر اعلام می کنیم که این کار نتیجه تلاش گروهی و همفکری جمعی اعضای تیم داده کاوان فونیکس بوده و همه به طور یکسان با همدلی سعی کردند پروژه را به بهترین نحو به سرانجام برسانند.

همچنین لازم می دانیم از سرکار خانم **لیلا نوربالا** که داده ها را در اختیار ما قرار دادند و همچنین جناب آقای **دکتر علیرضا وفایی صدر** که به عنوان مربی به تیم ما اعتماد کرده و این فرصت را به ما دادند و همچنین از ستاد علوم شناختی IPM نهایت تشکر را به عمل بیاوریم.

پایان