گزارش پروژه دوم درس داده‌کاوی

تحلیل احساسات موجود در توییت‌ها

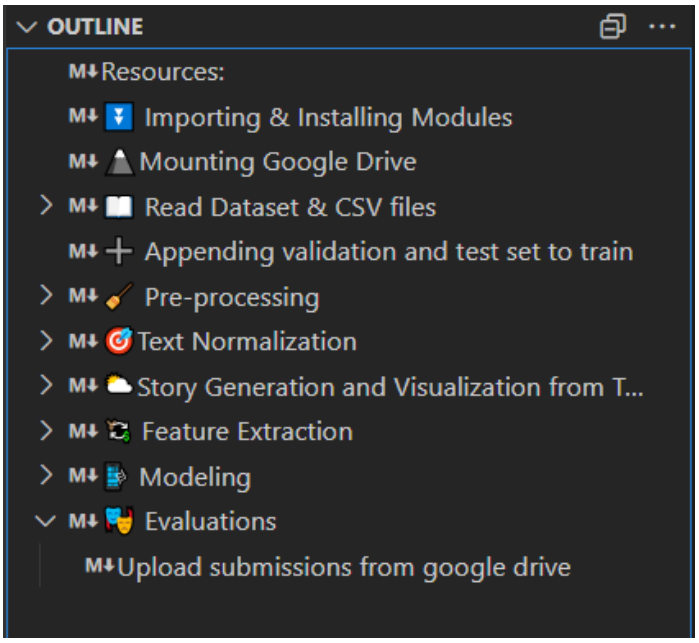
آیلین نائب‌زاده - 99522185

استاد درس: دکتر حسین رحمانی

نیم‌سال اول 1402-1403

- ## خواندن فایل‌ها و اطلاعات ورودی

برای پیاده‌سازی این پروژه همانند پروژه قبلی از محیط Google Colab و زبان Python استفاده شده‌است. همچنین برای درک راحت‌تر مراحل آن را به بخش‌های مجزا تقسیم کرده‌ام که می‌توانید در تصویر پائین مشاهده کنید.



اولین بخش پروژه مربوط به خواندن اطلاعات از فایل‌های ورودی می‌باشد. همانطور که در فایل اصلی توضیح داده شده‌است ما نیاز است که ابتدا اطلاعات مربوط به سه فایل را در برنامه خود ذخیره کنیم. داده‌های آموزشی، داده‌های تست و داده‌های صحت‌سنجی(valid). برای انجام این کار پس از خواندن اطلاعات از فایل‌های csv ورودی، اطلاعات هر فایل را بصورت جداگانه در قالب جدول درdataframe ذخیره می‌کنیم. برای خواندن اطلاعات موجود در داده‌هایی با نوع dataframe از توابع از پیش تعریف‌شده‌ای همانند .head و .info که در کتابخانه pandas موجود هستند می‌توانیم استفاده کنیم. همچنین از تابعی همانند .unique می‌توانیم استفاده کنیم تا مقادیر منحصربه‌فرد موجود در ستون 'sentiment' را شناسایی کنیم.

پیش از ادامه کار بااستفاده از تابع isnull و dropna مقادیر تهی را شناسایی و آن‌ها را داده‌ها حذف می‌کنیم.

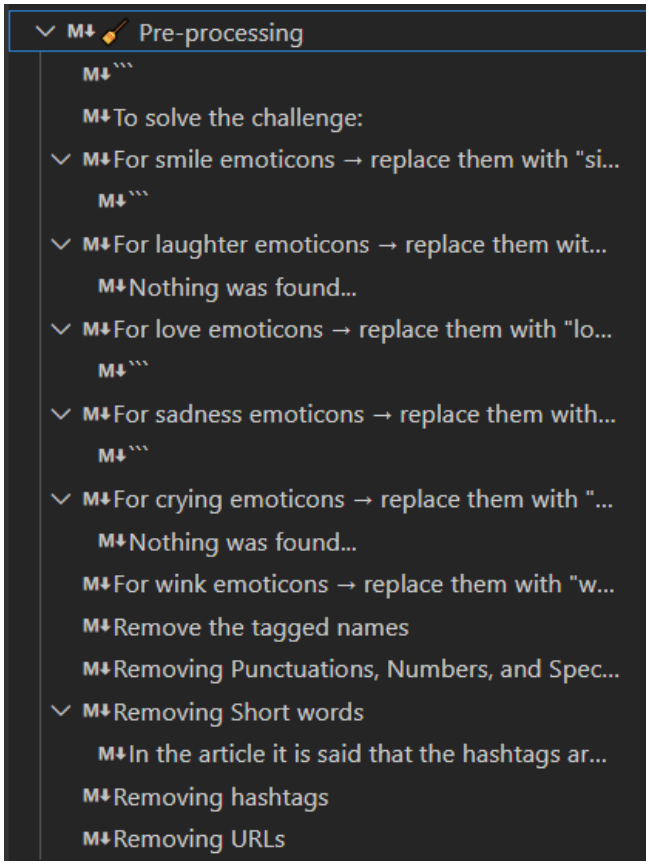
همچنین برای درک بهتر نسبت به توزیع داده‌ها و پراکندگی آن‌ها برای احساسات مختلف از نمایش داده‌ها برروی نمودار استفاده می‌کنیم. نمودار مربوط به این بخش را در تصویر زیر می‌توانید مشاهده کنید.

در ادامه داده‌های موجود از dataframe آموزشی، تست و صحت‌سنجی را در یک متغیر جمع‌آوری می‌کنیم تا مراحل پیش‌پردازش داده را نیاز نباشد که بصورت جداگانه انجام دهیم. اسم این متغیر در تمام طول برنامه combi می‌باشد. که درواقع تجمیع‌یافته تمامی داده‌های ما می‌باشد.

```
[19]  combi.info()

...   <class 'pandas.core.frame.DataFrame'>
      RangeIndex: 74996 entries, 0 to 74995
      Data columns (total 4 columns):
       #   Column         Non-Null Count  Dtype
      ---  ------         --------------  -----
       0   Tweet ID       74996 non-null  int64
       1   entity         74996 non-null  object
       2   sentiment      74996 non-null  object
       3   Tweet content  74996 non-null  object
      dtypes: int64(1), object(3)
      memory usage: 2.3+ MB
```

● پیش‌پردازش داده‌ها



در این مرحله درواقع سعی شده‌است محتوای موجود در ستون Tweet content را تمیزتر و آن را نرمال‌سازی نمود. از جمله کارهای انجام شده در این بخش می‌توان به مراحل زیر اشاره کرد:

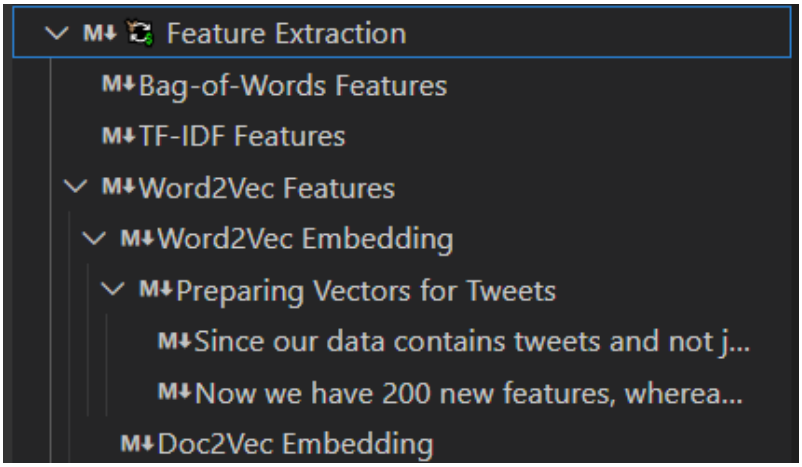
- جایگزینی نمادهای :)، :(و ... با کلمات نظیرشان مانند smile و sad(این کار بااستفاده از regex انجام شده‌است.)
- حذف تگ‌ها از محتوای توییت‌ها که با @ شروع شده‌اند.(این کار بااستفاده از regex انجام شده‌است.)
- حذف تمامی کاراکترهای خاص مانند اعداد و علائم نگارشی. (این کار بااستفاده از regex و تابع str.replace انجام شده‌است.)
- حذف تمامی کلماتی که طول آن‌ها کمتر از ۳ باشد.
- حذف هشتگ‌ها.(این کار بااستفاده از regex انجام شده‌است.)
- حذف آدرس‌ها و لینک‌های موجود.(این کار بااستفاده از regex انجام شده‌است.)

پس از پاک‌سازی متن نیاز است تا آن را نرمال‌سازی کنیم تا بتوانیم در مرحله بعدی ویژگی‌های آن را استخراج نمائیم.

برای انجام این کار ابتدا محتوای توییت‌ها را tokenize می‌کنیم به این‌صورت که برحسب فاصله کلمات را از یکدیگر جدا می‌کنیم. سپس بااستفاده از PortStemmer که در کتابخانه nltk موجود می‌باشد، کلمات را به ریشه‌های اصلی‌شان برمی‌گردانیم. و در نهایت مجددا آن را با یک فاصله به‌یکدیگر متصل می‌کنیم.

پس از انجام مراحل فوق می‌توانیم برای درک بهتر از کلمات پرتکرار از شبکه ابری واژگان استفاده کنیم. ابر واژگان مربوط به داده‌های ورودی را در تصویر زیر می‌توانید مشاهده کنید.

● انتخاب ویژگی‌ها



در این مرحله و برای آماده‌سازی داده‌ها برای مرحله آموزش و تست از الگوریتم‌های زیر استفاده شده‌است:

- Bag of Words 👉 using countVectorizer module in sklearn
- TF-IDF 👉 using TfidfVectorizer module in sklearn
- Word2Vec 👉 using Word2Vec module in gensim
- Doc2Vec 👉 using Doc2Vec module in gensim

در دو حالت اول با تعریف max_features = ۱۰۰۰ هزار ویژگی استخراج می‌شود ولی در دو حالت بعدی ۲۰۰ ویژگی استخراج می‌شود. درواقع در هنگام استفاده از هر یک از این توابع تعداد hyperparameter وجود دارد که می‌توانیم باتوجه به نیاز خود آن‌ها را مقداردهی کنیم.

● آموزش مدل

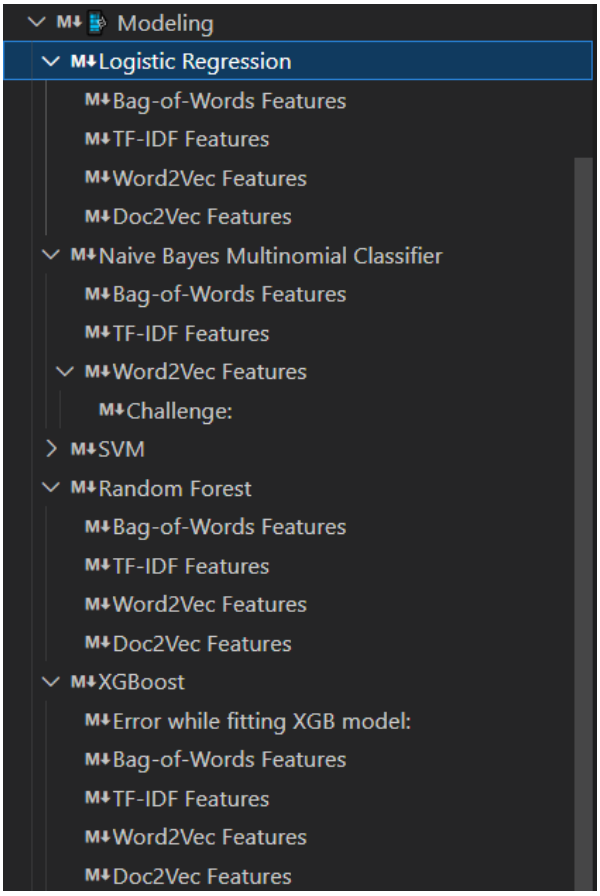
در این مرحله نیز از مدل‌های زیر استفاده شده‌است:

- Logistic Regression 👉 multinomial, 5000 max iterations, lbfgs solver
- Naive Bayes
- Random Forest 👉 11 random states, 400 estimators
- XGBoost 👉 6 as max depth, 1000 estimators

در ابتدا پیش از استفاده از هر یک از مدل‌ها ابتدا مقادیر ستون 'sentiment' را به اعداد ۱ تا ۴ مپ می‌کنیم. البته در حالت XGBoost نیاز است که از ۰ شروع شوند. سپس در هنگام استفاده از هر یک از الگوریتم‌های ذکر شده و بااستفاده از تابع .fit مدل را آموزش می‌دهیم. همچنین در هنگام آموزش، بصورت جداگانه هر یک از آرایه‌های مرحله مربوط به استراخ ویژگی را مورد آزمایش قرار می‌دهیم.

طبق خواسته مسئله نیز بااستفاده از تابع .predict و فراخوانی آن برروی آرایه داده‌های valid، از مدل خود برای پیش‌بینی استفاده می‌کنیم و نتایج را در فایل‌های csv در کنار مقادیر اصلی ذخیره می‌کنیم.

- ارزیابی مدل



در این مرحله بااستفاده از داده‌های تستی و معیارهایی همانند دقت و صحت، مدل‌های خود را مورد ارزیابی قرار می‌دهیم. و از توابع آماده موجود در کتابخانه sklearn همانند accuracy_score, precision_score, f1_score, recall_score و confusion_matrix استفاده می‌کنیم.

همانطور که در ادامه مشاهده می‌کنید، دو مدل Random Forest و XGBoost بهترین عملکرد را داشته‌اند.

نتایج مربوط به مدل‌های ذکر شده و ویژگی‌های گوناگون را در جداول زیر می‌توانید مشاهده کنید:

Logistic Regression	Accuracy	F1	Precision	Recall
BoW	64	63	64	63
TF-IDF	65	64	66	64
W2V	55	54	55	54
D2V	44	40	43	41

Naive Bayes	Accuracy	F1	Precision	Recall
BoW	61	60	60	60
TF-IDF	61	58	62	59

Random Forest	Accuracy	F1	Precision	Recall
BoW	89	88	89	88
TF-IDF	89	89	90	88
W2V	88	88	90	87
D2V	42	35	57	39

XGBoost	Accuracy	F1	Precision	Recall
BoW	88	87	88	87
TF-IDF	89	89	89	88
W2V	91	91	91	90
D2V	53	51	54	51

● نتیجه‌گیری

الف) همانطور که در بخش قبلی گفته شده بهترین مدل برای داده‌های فعلی Random Forest و XGBoost می‌باشند. عملکرد هر مدل وابسته به نوع داده‌ها و مسئله و همچنین تعریف hyperparameterها می‌باشد. ولی هر دو این مدل‌ها از نوع پیشرفته‌تر درخت تصمیم می‌باشند. همچنین می‌توان برای گروه‌بندی انواع داده‌های عددی و غیرعددی از آن‌ها استفاده کرد و می‌توانند الگوهای خطی و غیرخطی را تشخیص دهند. الگوریتم Random Forest از چند درخت تصمیم مستقل استفاده می‌کند و از قاعده majority voting پیروی می‌کند. الگوریتم XGBoost نیز بصورت متوالی از چند درخت تصمیم استفاده می‌کند و از الگوریتم Gradient Descent برای کمینه کردن میزان خطا استفاده می‌کند.

از دیگر مزایای این دو مدل می‌توان به این موضوع اشاره کرد که هر دو برای داده‌هایی از نوع نامتوازن مناسب هستند. نکته دیگر این است که نسبت به داده‌های outlier و noisy نیز حساس نیستند و عملکردشان دچار مشکل و خطای زیادی نمی‌شود.

همچنین هر دوی این الگوریتم‌ها نسبت به Overfit شدن مقاوم هستند.

ب)

Logistic Regression	Positive	Neutral	Negative	Irrelevant
BoW	67	56	77	42
W2V	58	57	69	31
TF-IDF	64	60	77	41
D2V	52	37	60	11

Naive Bayes	Positive	Neutral	Negative	Irrelevant
BoW	67	49	71	37
TF-IDF	70	47	75	16

Random Forest	Positive	Neutral	Negative	Irrelevant
BoW	93	88	92	98
W2V	96	89	95	88
TF-IDF	93	88	92	98
D2V	96	89	95	88

XGBoost	Positive	Neutral	Negative	Irrelevant
BoW	95	85	92	82
W2V	95	85	92	82
TF-IDF	95	85	92	82
D2V	95	85	92	82

طبق نتایج گزارش شده در جداول بالا، دو مدل XGBoost و Random Forest همانطور که برروی داده‌های تست بهترین عملکرد را داشتند، برای هر lable/feature/class بطور جداگانه نیز تعداد بیشتری را بصورت درست محاسبه کردند. و دلیل این موضوع را می‌توان به تعمیم‌پذیری بیشتر این دو مدل نسبت به دو مدل دیگر و رفتار آن‌ها نسبت به جلوگیری از Overfit شدن و حساس بودن به داده‌های noisy و Outlier دانست.