به نام خدا

گزارش کار تمرین اول

درس : یادگیری ماشین / Machine Learning

استاد : آقای دکتر رضوانیان

دانشجو : مهدی مهدیانی

شماره دانشجویی : ۴۰۲۱۳۳۴۰۴۱

آبان-آذر ماه ۱۴۰۲

ما در این پروژه از کتابخانه های sklearn’ scipy/Numpy/pandas استفاده کرده ایم که در فاز های مختلف به آن ها اشاره خواهیم کرد. در قدم اول ما دیتای اکسل را با کمک کتابخانه pandas در یک دیتا فریم ذخیره کردیم و یک EDA ساده را انجام دادیم. در این بخش متوجه شدیم که دیتای ما ۱۰۰ سطر دارد و دارای دو ستون جملات و زبان است/خوش بختانه دارای دیتای خالی (Null) نیستیم و ۵۰ درصد دیتاست جملات فارسی و مابقی عربی است. با این تفاسیر اولیه می توانیم کار خود را شروع کنیم . نکته مهم این است که لیبل ما در این پروژه ستون زبان است.
ابتدا لیستی از حروف عربی و فارسی را به صورت جداگانه می سازیم که حروف الفبای این دو زبان را شامل شود و سپس یک لیست از ترکیب این دو لیست به صورت غیر تکراری تهیه می کنیم که حکم فیچر ها یا ویژگی های ما را دارند.

مرحله ساخت بردار ویژگی ها (Feature Vectors)

در مرحله بعدی برای قسمت الف تا ت ما برای هر قسمت یک تابع جداگانه تعریف میکنیم که در زیر به آن ها می پردازیم.
قسمت الف : در این قسمت ما بردار ویژگی دودویی یا binary Bag of Character می خواهیم که تک تک جملات را می خوانیم و اگر کارکترهایی که در مرحله قبل به دست آوردیم در این جملات بود عدد ۱ را به آن نسبت می دهیم و اگر نبود عدد ۰ را نسبت می دهیم. در این مدل ما اهمیتی به تعداد حضور هر کارکتر نمی دهیم و صرفا حضور یا عدم حضور را بررسی می کنیم.
قسمت ب : در این قسمت ما بردار ویژگی وزن دار یا weighten bag of character می خواهیم که تک تک جملات را می خوانیم و تعداد کارکتر هایی که در هر جمله است را به دست می آوریم و تعداد آن را ذخیره می کنیم.
قسمت پ : در این قسمت ما بردار ویژگی هایی که در مرحله قبل به دست آورده ایم را نرمال میکنیم. روش کار ما به این صورت است که تعداد هر کارکتر در جمله را به دست می آوریم و سپس تقسیم بر اندازه طول جمله می کنیم و ذخیره می کنیم.
قسمت ت : در این قسمت با توجه به فرمول نرمال سازی به روش زد می خواهیم بردار ویژگی بسازیم. برای این امر با استفاده از کتابخانه scipy.stats و تابع zscore این کار را انجام میدهیم.
نکته ۱ : در آخر هر مرحله ما استفاده از کتابخانه pandas دیتافریم تولید شده را در یک فایل اکسل ذخیره می کنیم تا در مرحله بعد از آن استفاده کنیم.
نکته ۲: برای این قسمت ها تغییراتی را اعمال خواهیم کرد که در مرحله بعد به آن اشاره خواهیم کرد.

مرحله ساخت داده های آموزشی و تست (Train / Test Data):

در این مرحله ما باید برای هر کدام از بردار ویژگی هایی که در مراحل قبل ساختیم داده ی آموزشی و تست را مشخص کنیم .
طبق صورت سوال/داده های شماره ۱۰/۲۰/۳۰/...۱۰۰/ مجموعه تست ما هستند و ما بقی داده ها مجموعه آموزش ما خواهند بود. برای این مرحله ما مجموعه x1 و y1 ... x4 و y4 را برای هر دو بخش آموزش و تست می سازیم سپس آماده سازی های لازم را انجام می دهیم که بتوانیم آموزش را روی این مجموعه ها شروع کنیم.

مرحله آموزش مدل (Model Training):

در این مرحله با توجه به حضور KNeighborsClassifier که در کتابخانه scikit-learn قرار دارد آموزش را انجام خواهیم داد. این طبقه بند/امکان این را به ما می دهد که متریک های اقلیدسی و کسینوسی را روی پیاده کنیم. برای این کار کافیست به صورت زیر عمل کنیم:

KNeighborsClassifier(n_neighbors=k, metric=distance_metric)

بعد از آموزش می توانیم دقت آموزش را بر اساس accuracy_score که در کتابخانه sklearn.metrics وجود دارد بررسی کنیم و همچنین برای تعداد اشتباه می توانیم محاسبه کنیم که چند تا از لیبل های تست ما درست پیش بینی شده و چند تا اشتباه. تعداد اشتباه به تعداد کل برای ما اهمیت دارد.

جدول نتیجه شبیه سازی شماره ۱ :

پس از آموزش برای هر کدام از بردار ویژگی ها با احتساب تمامی حروف الفبا به نتیجه زیر رسیدیم :

	k	distance_metric	incorrect_predictions	accuracy
0	1	cosine	0	100.0
1	1	euclidean	0	100.0
2	3	cosine	0	100.0
3	3	euclidean	0	100.0
4	5	cosine	0	100.0
5	5	euclidean	0	100.0

جدول ۱ : نتیجه نهایی بر اساس binary BoW

	k	distance_metric	incorrect_predictions	accuracy
0	1	cosine	5	44.444444
1	1	euclidean	5	44.444444
2	3	cosine	5	44.444444
3	3	euclidean	5	44.444444
4	5	cosine	4	55.555556
5	5	euclidean	4	55.555556

جدول ۲ : نتیجه نهایی بر اساس weighten BoW

	k	distance_metric	incorrect_predictions	accuracy
0	1	cosine	0	100.0
1	1	euclidean	0	100.0
2	3	cosine	0	100.0
3	3	euclidean	0	100.0
4	5	cosine	0	100.0
5	5	euclidean	0	100.0

جدول ۳: نتیجه نهایی بر اساس Length Normalized BoW

	k	distance_metric	incorrect_predictions	accuracy
0	1	cosine	0	100.0
1	1	euclidean	0	100.0
2	3	cosine	0	100.0
3	3	euclidean	0	100.0
4	5	cosine	0	100.0
5	5	euclidean	0	100.0

جدول ۴: نتیجه نهایی بر اساس ZScore Normalized BoW

طبق نتایج به دست آمده/ما در همه جداول جز شماره ۲ دقت ۱۰۰ درصدی داریم و تمامی پیش بینی ما درست است. این می تواند به معنای Overfiting یا حفظ کردن داده های آموزشی باشد که دلایلی مثل :

1. نشت داده های تست به داده های آموزش
2. اندازه کوچک یا ناکافی مجموعه داده
3. ویژگی های بی ارتباط و ...

برای مورد اول بررسی های لازمه انجام گرفت که نشت داده ای رویت نشد.
مورد دوم می تواند تاثیر گذار باشد با توجه به تعداد کم داده های آموزشی
اما در مورد سوم/می توانیم راه کار هایی ارائه دهیم . مورد سوم وقتی رخ می دهد که ویژگی هایی در بردار ویژگی ها باشد که کاملا متمایز کننده کلاس های ما باشد . برای این کار راه های مختلفی را امتحان کرده که شاید تاثیر مثبتی داشته باشد .

قسمت اول :

ابتدا شروع به یکسان سازی بعضی از حروف کردیم. برای مثال حرف ي و ى/آ أ و ا/ة و ت و... را یکی در نظر گرفتیم. سپس مجدد مراحل قبل را با بردار ویژگی های جدید تست کردیم .

نتایج را مجددا بررسی کردیم :

	k	distance_metric	incorrect_predictions	accuracy
0	1	cosine	0	100.000000
1	1	euclidean	1	88.888889
2	3	cosine	0	100.000000
3	3	euclidean	0	100.000000
4	5	cosine	0	100.000000
5	5	euclidean	1	88.888889

جدول ۱ : نتیجه نهایی بر اساس binary BoW

	k	distance_metric	incorrect_predictions	accuracy
0	1	cosine	0	100.000000
1	1	euclidean	0	100.000000
2	3	cosine	0	100.000000
3	3	euclidean	1	88.888889
4	5	cosine	0	100.000000
5	5	euclidean	0	100.000000

جدول ۲ : نتیجه نهایی بر اساس weighten BoW

	k	distance_metric	incorrect_predictions	accuracy
0	1	cosine	0	100.0
1	1	euclidean	0	100.0
2	3	cosine	0	100.0
3	3	euclidean	0	100.0
4	5	cosine	0	100.0
5	5	euclidean	0	100.0

جدول ۳: نتیجه نهایی بر اساس Length Normalized BoW

	k	distance_metric	incorrect_predictions	accuracy
0	1	cosine	0	100.0
1	1	euclidean	0	100.0
2	3	cosine	0	100.0
3	3	euclidean	0	100.0
4	5	cosine	0	100.0
5	5	euclidean	0	100.0

جدول ۴: نتیجه نهایی بر اساس ZScore Normalized BoW

قسمت دوم :

در این قسمت ما متریک فاصله مینکوفسکی را هم به پروژه اضافه کردیم تا ببرسی کنیم خروجی ها به چه صورت خواهند بود . ازین پس در هر اجرا این معیار را هم حساب خواهیم کرد.

	k	distance_metric	incorrect_predictions	accuracy
0	1	cosine	0	100.000000
1	1	euclidean	1	88.888889
2	1	minkowski	1	88.888889
3	3	cosine	0	100.000000
4	3	euclidean	0	100.000000
5	3	minkowski	0	100.000000
6	5	cosine	0	100.000000
7	5	euclidean	1	88.888889
8	5	minkowski	1	88.888889

جدول ۱ : نتیجه نهایی بر اساس binary BoW

	k	distance_metric	incorrect_predictions	accuracy
0	1	cosine	0	100.000000
1	1	euclidean	0	100.000000
2	1	minkowski	0	100.000000
3	3	cosine	0	100.000000
4	3	euclidean	1	88.888889
5	3	minkowski	1	88.888889
6	5	cosine	0	100.000000
7	5	euclidean	0	100.000000
8	5	minkowski	0	100.000000

جدول ۲ : نتیجه نهایی بر اساس weighten BoW

	k	distance_metric	incorrect_predictions	accuracy
0	1	cosine	0	1.0
1	1	euclidean	0	1.0
2	1	minkowski	0	1.0
3	3	cosine	0	1.0
4	3	euclidean	0	1.0
5	3	minkowski	0	1.0
6	5	cosine	0	1.0
7	5	euclidean	0	1.0
8	5	minkowski	0	1.0

جدول ۳: نتیجه نهایی بر اساس Length Normalized BoW

	k	distance_metric	incorrect_predictions	accuracy
0	1	cosine	0	1.0
1	1	euclidean	0	1.0
2	1	minkowski	0	1.0
3	3	cosine	0	1.0
4	3	euclidean	0	1.0
5	3	minkowski	0	1.0
6	5	cosine	0	1.0
7	5	euclidean	0	1.0
8	5	minkowski	0	1.0

جدول ۴: نتیجه نهایی بر اساس ZScore Normalized BoW

قسمت سوم :

حال با توجه به توضیحاتی که درباره ویژگی های نامرتبط دادیم/تصمیم گرفتیم که بعضی از ویژگی ها مثل چ و ... را از بردار ویژگی ها حذف کنیم چرا که این ویژگی ها فقط در یک دسته از کلاس ما حاضر هستند و کاملا متمایز کننده کلاس ها هستند.
این ویژگی های امتحان من هستند :
چ پ گ ژ آ ي ة و ...

	k	distance_metric	incorrect_predictions	accuracy
0	1	cosine	2	0.777778
1	1	euclidean	3	0.666667
2	1	minkowski	3	0.666667
3	3	cosine	1	0.888889
4	3	euclidean	1	0.888889
5	3	minkowski	1	0.888889
6	5	cosine	1	0.888889
7	5	euclidean	1	0.888889
8	5	minkowski	1	0.888889

جدول ۱ : نتیجه نهایی بر اساس binary BoW

	k	distance_metric	incorrect_predictions	accuracy
0	1	cosine	0	100.000000
1	1	euclidean	1	88.888889
2	1	minkowski	1	88.888889
3	3	cosine	0	100.000000
4	3	euclidean	1	88.888889
5	3	minkowski	1	88.888889
6	5	cosine	0	100.000000
7	5	euclidean	0	100.000000
8	5	minkowski	0	100.000000

جدول ۲ : نتیجه نهایی بر اساس weighten BoW

	k	distance_metric	incorrect_predictions	accuracy
0	1	cosine	0	100.0
1	1	euclidean	0	100.0
2	1	minkowski	0	100.0
3	3	cosine	0	100.0
4	3	euclidean	0	100.0
5	3	minkowski	0	100.0
6	5	cosine	0	100.0
7	5	euclidean	0	100.0
8	5	minkowski	0	100.0

جدول ۳: نتیجه نهایی بر اساس Length Normalized BoW

	k	distance_metric	incorrect_predictions	accuracy
0	1	cosine	0	100.0
1	1	euclidean	0	100.0
2	1	minkowski	0	100.0
3	3	cosine	0	100.0
4	3	euclidean	0	100.0
5	3	minkowski	0	100.0
6	5	cosine	0	100.0
7	5	euclidean	0	100.0
8	5	minkowski	0	100.0

جدول ۴: نتیجه نهایی بر اساس ZScore Normalized BoW

قسمت پایانی :

در این قسمت با استفاده از Train_Test_Split از چهارچوب مشخص شده سوال خارج شده و مجموعه تست و آموزش رو به صورت زیر در نظر گرفته و مقدار های خطا را مجدد محاسبه کردیم.
توجه : از آخرین دیتافریم هایی که در مرحله قبل به دست آوردیم استفاده شده است .

به ازای ۷۰ درصد دیتاست مجموعه آموزش و ۳۰ درصد مجموعه تست :

	k	distance_metric	incorrect_predictions	accuracy
0	1	cosine	9	70.000000
1	1	euclidean	9	70.000000
2	1	minkowski	9	70.000000
3	3	cosine	4	86.666667
4	3	euclidean	5	83.333333
5	3	minkowski	5	83.333333
6	5	cosine	4	86.666667
7	5	euclidean	4	86.666667
8	5	minkowski	4	86.666667

جدول ۱ : نتیجه نهایی بر اساس binary BoW

	k	distance_metric	incorrect_predictions	accuracy
0	1	cosine	3	90.000000
1	1	euclidean	3	90.000000
2	1	minkowski	3	90.000000
3	3	cosine	3	90.000000
4	3	euclidean	2	93.333333
5	3	minkowski	2	93.333333
6	5	cosine	2	93.333333
7	5	euclidean	2	93.333333
8	5	minkowski	2	93.333333

جدول ۲ : نتیجه نهایی بر اساس weighten BoW

	k	distance_metric	incorrect_predictions	accuracy
0	1	cosine	4	86.666667
1	1	euclidean	7	76.666667
2	1	minkowski	7	76.666667
3	3	cosine	4	86.666667
4	3	euclidean	3	90.000000
5	3	minkowski	3	90.000000
6	5	cosine	2	93.333333
7	5	euclidean	2	93.333333
8	5	minkowski	2	93.333333

جدول ۳: نتیجه نهایی بر اساس Length Normalized BoW

	k	distance_metric	incorrect_predictions	accuracy
0	1	cosine	1	96.666667
1	1	euclidean	2	93.333333
2	1	minkowski	2	93.333333
3	3	cosine	1	96.666667
4	3	euclidean	1	96.666667
5	3	minkowski	1	96.666667
6	5	cosine	1	96.666667
7	5	euclidean	1	96.666667
8	5	minkowski	1	96.666667

جدول ۴: نتیجه نهایی بر اساس ZScore Normalized BoW

می توان این کار را با ۲۰ درصد مجموعه تست و ۸۰ درصد مجموعه آموزش هم انجام داد اما در کل بعد از این امتحان نتیجه گرفتم که اگر تعداد دیتای تست را بیشتر کنیم در این مثال احتمال ۱۰۰ درصد شدن جواب ما و overfit شدن کم تر خواهد شد.
البته می توان نمودار learning curve را رسم کرد تا بررسی شود آیا واقعا overfit شده ایم یا خیر.

نتیجه گیری :

در این تمرین ما به بررسی الگوریتم KNN پرداختیم. ابتدا دیتاست مورد نظر را بررسی کردیم و سپس به ساخت بردار ویژگی ها پرداختیم. بعد از تعیین مجموعه تست و آموزش مدل را آموزش دادیم و دقت و تعداد خطا را تعیین کردیم. به مساله Overfit برخوردیم و تلاش کردیم با امتحان راه های مختلف این مشکل را برطرف کنیم. در پایان به این رسیدیم که اگر بردار ویژگی های مناسب را انتخاب نکنیم و تعداد داده های دیتاست ما کم باشد احتمال overfit شدن بیشتر می شود .
اگر overfit شدیم می توانیم از تکنیک های active learning استفاده کنیم که ازین مشکل خارج شویم.
نکته : با توجه به محدودیت وقت نتوانستم راجع به active learning مطالعه کنم و در این سوال پیاده سازی کنم .