((به نام خدا))

نام دانشجو:

پویان حسابی

شماره دانشجویی:

۹۷۳۱۱۲۲

گزارش پروژه پایانی درس داده کاوی

استاد درس: دکتر ناظرفرد

بهار ۱۴۰۱

"پروژه پایانی درس داده کاوی"

مقدمه

در این پروژه قصد داریم با استفاده از تکنیک های classification و کتابخانه XGBoost مشخص کنیم که فردی با مشخصات خاص، دارای دیابت می‌باشد یا خیر. مشخصه ها و داده های مربوط به پروژه در فایلی قرار دارد که در قالب زیر می‌باشد، هر کدام از فیلد ها در تایپ های عددی، بولین(به شکل ۰ و ۱) و رشته که کتگوریکال است می‌باشند.

	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W
1		Diabetes_I	HighBP	High Chole	Cholesterc	BMI	Smoker	Stroke	HeartDise	Physical A	Fruits	Veggies	Heavy Alc	Any Health	No Doctor	General He	Mental He	Physical He	Difficulty \	Sex	Age	Education	Income
2	2456	0	0	1	1	24	0	0	0	1	1	0	0	1	1	Very Low	30	30	0	male	1	Cat6	Cat6

این پروژه در ۳ فاز انجام می‌شود که در ادامه هر کدام از آنها توضیح داده می‌شوند.

مرحله پیش پردازش

قبل از اینکه پردازش اصلی و ساخت مدل را انجام دهیم، نیاز به پیش پردازش داریم تا دیتای ما به اصطلاح تمیز شود. از طرفی پیش پردازش دقت مدل را به شدت افزایش می‌دهد و یک گام الزامی می‌باشد.

در ابن مرحله ابتدا ستون سمت چپ که اضافه است را پاک می‌کنیم، سپس تمام white space ها را به کاراکتر "_" تبدیل می‌کنیم تا کتابخانه XGBoost به مشکل نخورد. بعد از آن داده هایی که نال یا خالی هستند را پر می‌کنیم به این روش که اگر عدد یا بولین(صفر یا یک) بود مقدار میانه آن ستون را جایگزین مقدار خالی می‌کنیم، و اگر داده های categorized باشد مقداری در ستون که بیشترین تکرار را دارد.

بعد از آن داده هایی که به اصطلاح اشتباه می‌باشند یا خارج از عرف هستند اصلاح می‌کنیم. به طور مثال تقریبا BMI بالای ۸۰ غیر ممکن است آن را کاهش می‌دهیم یا اگر فیلدی در بولین ۲ باشد باید تغییر کند.

در گام بعدی داده ها را نرم می‌کنیم که دسته های کمتری داشته باشند، به طور مثال age را به دسته های ۱۰ تایی تقسیم کرده و BMI را در بازه های استاندارد واقعی BMI جایگزین می‌کنیم.

مرحله بعد دسته بندی داده های categorical است که با استفاده از one-hot-encoding آنها را به دسته های مختلف تقسیم می‌کنیم و جای آن دسته ها صفر و یک قرار می‌گیرد.

در آخر ستون Diabetes_binary را به عنوان labeled_diabetes جدا کرده و از data_frame حذف می‌کنیم.

ساخت مدل طبقه بند

در این قسمت با استفاده از کتابخانه XGBoost دسته بندی را انجام می‌دهیم به این شکل که قسمتی از داده ها برای آموزش و قسمتی دیگر برای تست به نسبت ۷۵و ۲۵ درصد در نظر می‌گیریم.

پارامتر های زیر را به صورت ورودی به این تابع می‌دهیم:

XGBClassifier(Learning_rate=0.1, Max_depth=4, N_estimator=200, Subsample=0.5, Colsample_bytree=1, Random_seed=123, Eval_metric='auc', Verbosity=1)

بعد از ساخت مدل و استفاده از تابع های predict و fit، مدل را به تابع model_evaluation می‌دهیم تا دقت، صحت، پوشش و ماتریس درهم ریختگی را نشان دهد.

```
This is for  train
Confusion Matrix:  [[7005 1783]
 [1281 7604]]
Accuracy:  0.8266281898941888
precision:  0.8266281898941888
recall:  0.8454018826937002
This is for  test
Confusion Matrix:  [[18547  8011]
 [ 6107 20354]]
Accuracy:  0.7337181010581112
precision:  0.7337181010581112
recall:  0.7522917173683784
```

تنظیم هایپر پارامتر

پارامتر های زیادی در ساخت مدل تاثیر گذار هستند، که هر کدام از آنها در عملکرد و سرعت مدل اثرگذار است. در این بخش قصد داریم بهترین مقادیر را برای پارامتر ها بیابیم به این شکل که مدل حداکثر عملکرد را داشته باشد.

در واقع با ترکیب پارامتر های زیر این کار را انجام می‌دهیم:

learning_rates = [0.02, 0.05, 0.1, 0.3], max_depths = [2, 3, 4]

n_estimators = [100, 200, 300], colsample_bytrees = [0.8, 1]
hyper_parameters = [learning_rates, max_depths, n_estimators, colsample_bytrees]

سپس مدل و پارامتر ها را به تابع GridSearchCV داده و با تکنیک سه نقطه جداسازی و scoring='roc_auc' بهترین پارامتر ها و دقت، صحت، پوشش و ماتریس درهم ریختگی را نمایش می‌دهیم:

```
GridSearchCV(cv=StratifiedKFold(n_splits=3, random_state=7, shuffle=True),
             estimator=XGBClassifier(base_score=None, booster=None,
                                     callbacks=None, colsample_bylevel=None,
                                     colsample_bynode=None,
                                     colsample_bytree=None,
                                     early_stopping_rounds=None,
                                     enable_categorical=False,
                                     eval_metric='auc', gamma=None, gpu_id=None,
                                     grow_policy=None, importance_type=None,
                                     interaction_...
                                     max_leaves=None, min_child_weight=None,
                                     missing=nan, monotone_constraints=None,
                                     n_estimators=100, n_jobs=None,
                                     num_parallel_tree=None, predictor=None,
                                     random_state=None, reg_alpha=None,
                                     reg_lambda=None, ...),
             n_jobs=-1,
             param_grid={'colsample_bytree': [0.8, 1],
                         'learning_rate': [0.02, 0.05, 0.1, 0.3],
                         'max_depth': [2, 3, 4],
                         'n_estimators': [100, 200, 300]},
             scoring='roc_auc')
```

```
Best parameters:  {'colsample_bytree': 0.8, 'learning_rate': 0.05, 'max_depth': 4, 'n_estimators': 300}
```

ارزیابی مدل:

```
Best parameters:  {'colsample_bytree': 0.8, 'learning_rate': 0.05, 'max_depth': 4, 'n_estimators': 300}
This is for  test
Confusion Matrix:  [[18491  8067]
 [ 6407 20054]]
Accuracy:  0.7270035270374772
precision:  0.7270035270374772
recall:  0.7426700939834525
This is for  train
Confusion Matrix:  [[6960 1828]
 [1365 7520]]
Accuracy:  0.8193289198211962
precision:  0.8193289198211962
recall:  0.836036036036036
```

مشاهده می‌شود که مقادیر Accuracy, Precision, Recall همگی در مرحله train نسبت به حالت قبل، به مراتب بهتر شده و عملا مدل عملکرد بهتری پیدا کرده است.