باسمه تعالی

گزارش کار تمرین اول درس یادگیری ماشین

استاد درس : سرکار خانم دکتر ساجدی

نام دانشجو : ایمان کیانیان

شماره دانشجویی : ۶۱۰۳۰۰۲۰۳

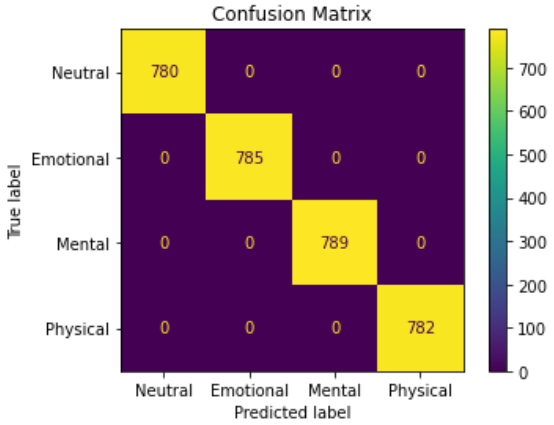
۱- مقدمه

در این تمرین، با استفاده از روش های متنوع همچون **SVM**، **XGBoost**، **Random Forest** و **DecisionTree** یک مسئله دسته بندی را مدل کردیم . داده های ما شامل ۵۳۳ ستون و ۴۴۸۰ سطر هستند که بیانگر وجود ۵۳۳ فیچر و ۴۴۸۰ سمپل است. همچنین یک بردار ۴۴۸۰ تایی برای لیبل های داده ها داریم. در ابتدا داده ها را به ۲ بخش داده های آموزشی و داده های تست تقسیم کردیم. ۷۰ درصد از داده های ما یعنی ۳۱۳۶ سمپل مربوط به داده های آموزشی و ۳۰ درصد داده ها، معادل با ۱۳۴۴ سمپل مربوط به داده های تست است. با استفاده از الگوریتم های ذکر شده و داده های وارد شده، مدل ها را آموزش داده و سپس تست کردیم و نتایج را به تفکیک در ادامه گزارش خواهیم دید. در تمام مدل های این درخت از **random_state=0** استفاده کردیم به دلیل اینکه میخواستیم در هر بار اجرا دقت واحدی بگیریم و هیچ چیز تصادفی نباشد.

۲- درخت تصمیم – Decision tree:

در این روش با استفاده از کتابخانه **sklearn**، ابتدا داده های آموزشی را برای آموزش به یک مدل **decisiontree** با معیار **انتروپی** دادیم دیگر پارامتر های ما پارامتر های **default** کتابخانه **sklearn** بود که میتوانید در **[1]**[۱] مشاهده کنید. یادگیری انجام شد و دقت روی داده های آموزشی برابر ۱۰۰ درصد بود که درصد بسیار بالایی است. نشانگر این است که داده های ما به راحتی توسط **decision tree** و فیچر ها جدا شده اند که این موضوع خوبی است . اما احتمالا چون **pruning** یا **feature selection** انجام نداده ایم ممکن است گرفتار **overfitting** شویم. برای اینکار داده های تست را به مدل آموزش داده شده می دهیم. نباید دقت پایینتری نسبت به داده های آموزشی بدهد وگرنه متوجه **overfitting** میشویم.

Confusion matrix برای داده های تست به شکل زیر است که نشان میدهد حتی ۱ غلط هم وجود ندارد:



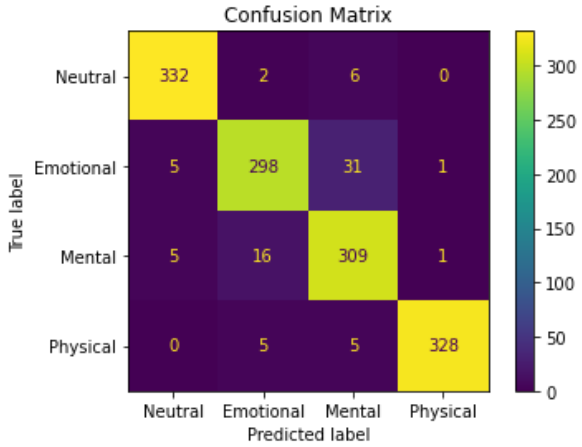
مقادیر دقت ها با روش های مختلف مثل **precision** ، **recall**، **f1-score** روی داده های آموزشی به صورت زیر است:

[۱] [1]. https://scikit-learn.org/stable/modules/generated/sklearn.tree.DecisionTreeClassifier.html

	Precision	recall	f1-score	support
Neutral	1.00	1.00	1.00	780
Emotional	1.00	1.00	1.00	785
Mental	1.00	1.00	1.00	789
Physical	1.00	1.00	1.00	782
accuracy			1.00	3136
macro avg	1.00	1.00	1.00	3136
weighted avg	1.00	1.00	1.00	3136

سپس با استفاده از داده های تست مدل خودمان را امتحان میکنیم و دقت ما برابر تقریبا ۹۴ درصد خواهد بود که نشانگر این است کمی مدل ما اورفیت شده است. میتوانیم با استفاده از روش هایی مثل **pruning** یا **feature extraction** یا **feature selection** کمی از احتمال اورفیت کم کنیم.

Confusion matrix را در عکس زیر برای داده های تست میبینید:



مقادیر دقت ها با روش های مختلف مثل **precision** ، **recall** ، **f1-score** روی داده های تست به صورت زیر است:

	precision	recall	f1-score	support
Neutral	0.97	0.98	0.97	340
Emotional	0.93	0.89	0.91	335
Mental	0.88	0.93	0.91	331
Physical	0.99	0.97	0.98	338
accuracy			0.94	1344
macro avg	0.94	0.94	0.94	1344
weighted avg	0.94	0.94	0.94	1344

درخت تصمیم رسم شده در کنار فایل پروژه ذخیره شده است که میتوانید مشاهده کنید.

سپس برای استفاده کردن از **k-fold** ابتدا داده ها را **shuffle** کرده، سپس با استفاده از **5-fold-cross-validation** داده های آموزشی را به ۵ بخش مساوی تقسیم میکنیم. در هر مرحله ۴ قسمت از این ۵ قسمت **training set** و یک قسمت **validation set** خواهد بود. سپس دقت ها را در هر بار تکرار گرفته جمع کرده و تقسیم بر ۵ میکنیم. دقت تقریبا ۹۰ درصد (**0.8960584727597801**) خواهد بود.

جدول نتایج بدست آمده در مرحله اول این تمرین را مشاهده میکنید :

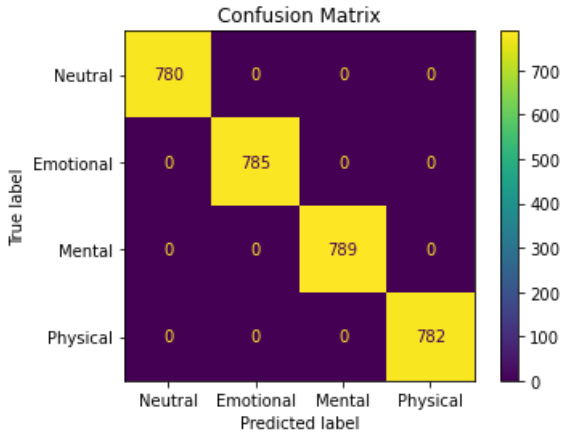
Data	Accuracy	Precision	Recall	f-measure
Training Data	*1.0*	*1.0*	*1.0*	*1.0*
Test Data	*0.9427* *(Overfitting)*	*0.94*	*0.94*	*0.94*

۳- جنگل تصادفی – Random forest

در این قسمت با استفاده از مدل **random forest** داده های آموزشی را گرفتیم و مدل را آموزش دادیم. ۱۰۰ درخت تصمیم (**n_estimators**) در این مدل استفاده شد که به عنوان داور ها ایفای وظیفه میکنند. پارامتر ها به صورت زیر است:

```
class sklearn.ensemble.RandomForestClassifier(n_estimators=100, *, criterion='entropy', max_depth=None,
min_samples_split=2, min_samples_leaf=1, min_weight_fraction_leaf=0.0, max_features='auto', max_leaf_nodes=None,
min_impurity_decrease=0.0, bootstrap=True, oob_score=False, n_jobs=None, random_state=0, verbose=0, warm_start=False,
class_weight=None, ccp_alpha=0.0, max_samples=None)
```

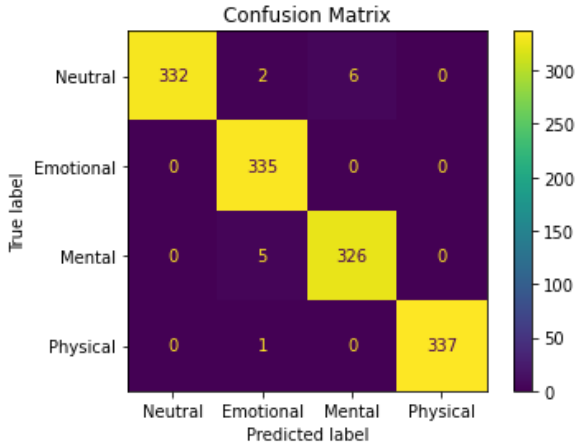
دقت بر روی مجموعه ی آموزشی پس از آموزش برابر با ۱۰۰ درصد است. احتمال **overfitting** وجود دارد. اما چون **random forest** استفاده کردیم احتمال **overfitting** نسبت به **decision tree** خیلی کمتر خواهد شد. در ادامه درباره اینکه آیا **overfitting** اتفاق افتاده است یا خیر صحبت خواهیم کرد. **Confusion matrix** برای داده های آموزشی به صورت زیر است:



همچنین دقت با روش های مختلف مثل **precision**، **recall**، **f1-score** روی داده های آموزشی به صورت زیر است:

	precision	recall	f1-score	support
Neutral	1.00	1.00	1.00	780
Emotional	1.00	1.00	1.00	785
Mental	1.00	1.00	1.00	789
Physical	1.00	1.00	1.00	782
accuracy			1.00	3136
macro avg	1.00	1.00	1.00	3136
weighted avg	1.00	1.00	1.00	3136

سپس داده های تست را برای ارزیابی به مدل آموزش داده شده ی **random forest** دادیم. مشاهده کردیم که دقت برای داده های تست برابر تقریبا ۹۹ درصد است (**0.9895833333333334** درصد). مقدار کمی **overfitting** داریم که اصلا به چشم نمی آید. این خاصیت **random forest** است که از اورفیت جلوگیری میکند به دلیل وجود تعداد بالای درخت های تصمیم و چون نظر ها جامعیت پیدا میکند پس احتمال اورفیت کم خواهد شد. **Confusion matrix** برای داده های تست به صورت زیر خواهد بود:



دقت با روش های مختلف مثل **precision**، **recall**، **f1-score** روی داده های تست به صورت زیر است:

	precision	recall	f1-score	support
Neutral	1.00	0.98	0.99	340
Emotional	0.98	1.00	0.99	335
Mental	0.98	0.98	0.98	331
Physical	1.00	1.00	1.00	338
accuracy			0.99	1344
macro avg	0.99	0.99	0.99	1344
weighted avg	0.99	0.99	0.99	1344

سپس برای استفاده کردن از **k-fold** ابتدا داده ها را **shuffle** کرده، سپس با استفاده از **5-fold-cross-validation** داده های آموزشی را به ۵ بخش مساوی تقسیم میکنیم. در هر مرحله ۴ قسمت از این ۵ قسمت **training set** و یک قسمت **validation set** خواهد بود. سپس دقت ها را در هر بار تکرار گرفته جمع کرده و تقسیم بر ۵ میکنیم. دقت تقریبا ۹۸ درصد (**0.9821473196598909**) خواهد بود.

جدول نتایج بدست آمده در مرحله دوم این تمرین را مشاهده میکنید :

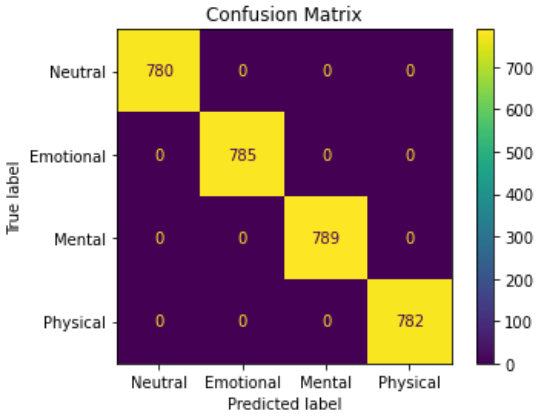
Data	Accuracy	Precision	Recall	f-measure
Training Data	*1.0*	*1.0*	*1.0*	*1.0*
Test Data	*0.9895* *(a bit Overfitting But it's ok…)*	*0.99*	*0.99*	*0.99*

۴- تقویت شدید گرادیان – XGBoost

در این روش یک مدل **XGBoost** ساختیم. سپس آن را با داده های آموزشی **train** کردیم. و روی داده های آموزشی و تست ، دقت را اندازه گیری کرده و در ادامه نتایج را اعلام خواهیم نمود. در این قسمت از کتابخانه **sklearn** استفاده نکردیم بلکه از کتابخانه مجزای **xgboost** استفاده کردیم که پارامتر های آن به صورت زیر است:

```
XGBClassifier(base_score=0.5, booster='gbtree', colsample_bylevel=1,
              colsample_bynode=1, colsample_bytree=1, enable_categorical=False,
              gamma=0, gpu_id=-1, importance_type=None,
              interaction_constraints='', learning_rate=0.300000012,
              max_delta_step=0, max_depth=6, min_child_weight=1, missing=nan,
              monotone_constraints='()', n_estimators=100, n_jobs=8,
              num_parallel_tree=1, objective='multi:softprob', predictor='auto',
              random_state=0, reg_alpha=0, reg_lambda=1, scale_pos_weight=None,
              subsample=1, tree_method='exact', validate_parameters=1,
              verbosity=None)
```

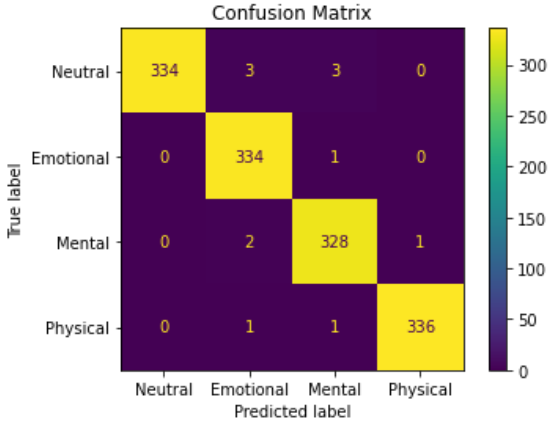
دقت این مدل روی داده های آموزشی برابر ۱۰۰ درصد و دقت روی داده ی تست بیش از ۹۹ درصد است که میبینیم به علت ماهیت **xgboost** از میزان **overfitting** نسبت به **decision tree** و **random forest** کم تر شده است. **Confusion matrix** را برای داده ی آموزشی مشاهده میکنید:



دقت با روش های مختلف مثل **precision**، **recall**، **f1-score** روی داده های آموزشی به صورت زیر بدست آوردیم:

	precision	recall	f1-score	support
Neutral	1.00	1.00	1.00	780
Emotional	1.00	1.00	1.00	785
Mental	1.00	1.00	1.00	789
Physical	1.00	1.00	1.00	782
accuracy			1.00	3136
macro avg	1.00	1.00	1.00	3136
weighted avg	1.00	1.00	1.00	3136

Confusion matrix برای داده های تست به صورت زیر است:



دقت با روش های مختلف مثل **precision**، **recall**، **f1-score** روی داده های تست به صورت زیر است:

	precision	recall	f1-score	support
Neutral	1.00	0.98	0.99	340
Emotional	0.98	1.00	0.99	335
Mental	0.98	0.99	0.99	331
Physical	1.00	0.99	1.00	338
accuracy			0.99	1344
macro avg	0.99	0.99	0.99	1344
weighted avg	0.99	0.99	0.99	1344

سپس برای استفاده کردن از **k-fold** ابتدا داده ها را **shuffle** کرده، سپس با استفاده از **5-fold-cross-validation** داده های آموزشی را به ۵ بخش مساوی تقسیم میکنیم. در هر مرحله ۴ قسمت از این ۵ قسمت **training set** و یک قسمت **validation set** خواهد بود. سپس دقت ها را در هر بار تکرار گرفته جمع کرده و تقسیم بر ۵ میکنیم. دقت تقریبا ۹۹ درصد (**0.9859694836396145**) خواهد بود.

جدول نتایج بدست آمده در مرحله سوم این تمرین را مشاهده میکنید :

Data	Accuracy	Precision	Recall	f-measure
Training Data	*1.0*	*1.0*	*1.0*	*1.0*
Test Data	*0.9910* *(a bit Overfitting But it's ok.)*	*0.99*	*0.99*	*0.99*

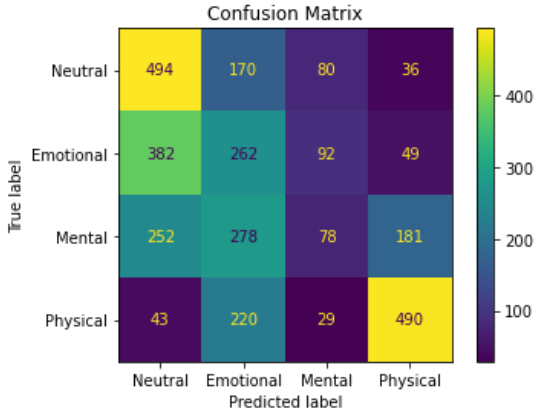
در این ۳ قسمت ، مشاهده شد که مدل های تست شده (**decision tree**، **random forest** و **xgboost**) توانایی مدلسازی داده های ما را دارند و میتوانند به راحتی تابع هدف را تخمین بزنند اما دچار مشکل **overfitting** میشویم. مشاهده کردیم که به ترتیب که روش ها را تغییر میدادیم به علت ماهیت بهتر از اورفیت شدن تا حدی جلوگیری میکرد و مدل را محدود میکرد(به علت **ensemble** بودن). در ادامه با مدل **SVM** کار خواهیم کرد که نشان میدهد این روش توانایی مدل کردن این داده ها را ندارد.

۵- ماشین بردار پشتیبان – SVM

در این روش با استفاده یک مدل **SVM** از نوع **OneVsRest** ، هدف ما این بود که داده های خود را آموزش داده و عمل دسته بندی را انجام دهیم. با توجه به اینکه **SVM** با **support vector** ها کار میکند و تقریبا به داده های خیلی زیادی احتیاج ندارد، انتظار داریم اگر داده ها قابل جدا شدن توسط خطوط باشند دقت بسیار بالایی بگیریم. البته لازم به ذکر است پارامتر های تنظیم شده در این مدل **SVM** به صورت زیر است:

```
class sklearn.svm.LinearSVC(penalty='l2', loss='squared_hinge', *, dual=True, tol=0.0001, C=1.0, multi_class='ovr',
fit_intercept=True, intercept_scaling=1, class_weight=None, verbose=0, random_state=0, max_iter=1000)
```

کرنل های مختلفی نیز آزمایش کردیم و بهترین کرنل **rbf** بود. بعد از انجام مدلسازی و آموزش ، مدل را برای داده های آموزشی تست کردیم. دقت روی داده های آموزشی حدود ۴۲ درصد (**0.4221938775510204**) است! یعنی عملا چون **SVM** نیاز به داده ی زیادی ندارد پس **SVM** روش کارایی برای دسته بندی این داده ها نیست چون داده ها **linear separable** نیستند. اگر روی داده های آموزشی دقت ۱۰۰ و روی داده های تست کم بود میگفتیم قطعا اورفیت شده است. اما حال باید ادعا کنیم که مدل ما دچار **underfitting** شده است. **Confusion Matrix** برای داده های آموزشی به صورت زیر است:



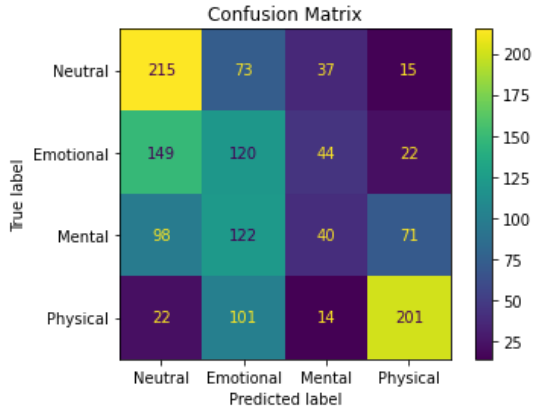
از تحلیل این جدول چند چیز قابل مشاهده است. داده های کلاس **physical** و **Neutral** به راحتی توسط یک خط از هم جدا میشوند. داده های کلاس **mental** با همه ی کلاس ها تداخل زیادی دارد! در واقع **SVM** نتوانسته بین **Mental** و بقیه کلاس ها جدا ساز ایجاد کند. مشکل اصلی مرز بین **Mental** و **Neutral** و **Emotional** و **physical** است که بسیار پیچیده جدا خواهند شد یا اصلا بصورت خطی قابل جداسازی نیستند. بنابراین مشخص کردن مرز بین اینها خیلی سخت است.

پس دیدیم که به راحتی میتوان با استفاده از **Confusion matrix** وضعیت داده ها را تحلیل کرد.

دقت با روش های مختلف مثل **precision**، **recall**، **f1-score** روی داده های آموزشی به صورت زیر است:

	precision	recall	f1-score	support
Neutral	0.42	0.63	0.51	780
Emotional	0.28	0.33	0.31	785
Mental	0.28	0.10	0.15	789
Physical	0.65	0.63	0.64	782
accuracy			0.42	3136
macro avg	0.41	0.42	0.40	3136
weighted avg	0.41	0.42	0.40	3136

سپس این مدل را روی داده های تست آزمایش کردیم. انتظار نتیجه خوبی نداشتیم و همینطور هم شد. دقت تقریبا ۴۳ درصد (**0.42857142857142855**) که البته دقت بیشتری از داده آموزشی گرفتیم و این نشان میدهد ، **SVM** ظرفیت و قدرت مدلسازی بیشتر از این برای این داده ها ندارد. **Confusion matrix** را برای داده های تست مشاهده می کنید:



و همینطور دقت با روش های مختلف مثل **precision**، **recall**، **f1-score** روی داده های تست به صورت زیر است:

	precision	recall	f1-score	support
Neutral	0.44	0.63	0.52	340
Emotional	0.29	0.36	0.32	335
Mental	0.30	0.12	0.17	331
Physical	0.65	0.59	0.62	338
accuracy			0.43	1344
macro avg	0.42	0.43	0.41	1344
weighted avg	0.42	0.43	0.41	1344

سپس برای استفاده کردن از **k-fold** ابتدا داده ها را **shuffle** کرده، سپس با استفاده از **5-fold-cross-validation** داده های آموزشی را به ۵ بخش مساوی تقسیم میکنیم. در هر مرحله ۴ قسمت از این ۵ قسمت **training set** و یک قسمت **validation set** خواهد بود. سپس دقت ها را در هر بار تکرار گرفته جمع کرده و تقسیم بر ۵ میکنیم. دقت تقریبا ۳۷ درصد (**0.353007954164508**) خواهد بود که دقت اصلا خوبی نیست. جدول نتایج بدست آمده در مرحله چهارم این تمرین را مشاهده میکنید :

Data	Accuracy	Precision	Recall	f-measure
Training Data	*0.4222* *(!!underfitting!!)*	*0.41*	*0.42*	*0.40*
Test Data	*0.4286*	*0.42*	*0.43*	*0.41*

۶- نتیجه گیری

در این قسمت نتیجه گیری های کلی از این تمرین را ذکر می کنیم. همانطور که در قبل گزارش شد، **decision tree** قابلیت مدل کردن داده های این مسئله را دارد اما گرفتار **overfitting** می شود و باید از رویکرد های اجتناب از **overfitting** همانند **pruning** یا **feature selection** و یا **feature extraction** و ... استفاده کرد اما چون هدف در این تمرین استفاده از این روش ها نبوده، میتوانیم از این قضیه صرف نظر کنیم.

با توجه به اینکه یکی از خواص روش های **ensemble** مثل **random forest** و **xgboost** این است که احتمال **overfitting** را کم کرده و **generalization** را برای ما به ارمغان می آورد ، همانند انتظار چنین نتیجه ای گرفتیم و نتایج برای داده های تست خیلی رشد کرد.

سپس از **SVM** استفاده کردیم و به این نتیجه رسیدیم که کلاس های ما **linear separable** نیستند و نمیتوان با **SVM** جداسازی خوبی انجام داد . در واقع برای چنین مسئله ای **SVM** مناسب نیست. جدول کلی دقت و نتایج گرفته شده برای داده های آموزشی به صورت زیر است:

model	Value of Parameters	Accuracy	Precision	Recall	f-measure
Decision Tree	*Criterion : entropy* *random_state = 0* *Others : Click*	*1.0*	*1.0*	*1.0*	*1.0*
Random Forest	*n estimators : 100* *Criterion : entropy* *random_state = 0* *Others : Click*	*1.0*	*1.0*	*1.0*	*1.0*
XGBoost	*Objective = multi:softprob* *random_state = 0* *Others : Click*	*1.0*	*1.0*	*1.0*	*1.0*
SVM	*random_state = 0* *Default : Click and Click*	*0.4222*	*0.41*	*0.42*	*0.40*

و جدول کلی دقت و نتایج گرفته شده برای داده های تست به صورت زیر است:

model	Value of Parameters	Accuracy	Precision	Recall	f-measure
Decision Tree	*Criterion : entropy* *Others : Click*	*0.9427*	*0.94*	*0.94*	*0.94*
Random Forest	*n_estimators : 100* *Criterion : entropy* *Others : Click*	*0.9895*	*0.99*	*0.99*	*0.99*
XGBoost	*Objective : multi:softprob* *random_state : 0* *Others : Click*	*0.9910*	*0.99*	*0.99*	*0.99*
SVM	*Kernel : rbf* *Default : Click and Click*	*0.4222*	*0.41*	*0.42*	*0.40*

تحلیل های لازم درباره هر نتیجه در هر قسمت مربوطه توضیح داده شده است. البته میشد پارامتر های بیشتری برای هر مسئله امتحان کرد . اما بهترین دقت ها به این شکل حاصل شده است. البته دقت داریم که این نتایج با نتایجی که درون مقاله گرفته شده است متفاوت خواهد بود چون در آنجا هدف **feature selection** بوده است اما در اینجا ما **feature selection** انجام ندادیم و فقط اهداف ما استفاده از این روش ها برای دسته بندی داده هاست.

دقت داشته باشید ، چون دوباره فایل **ipynb** چندین بار اجرا شده است و بعد از گزارش گیری هم اجرا شده است ممکن است دقت ها کمی با دقت هایی که در گزارش نوشتیم متفاوت باشد. اما کلیت به صورت گزارش هایی است که در این فایل آمده است.

پایان.