

مهدی فقهی

۴۰۱۷۲۲۱۳۶

پروژه دوم شناسایی الگو

Diagnosis_of_epilepsy:

در سوال اول از ما خواسته شده بود براساس داده‌های داده شده به کمک **AdaBoostClassifier** و به شکل **Cross Validation n fold** و به ازای n برابر با 5, 7, 10 دقت مدل را بدست آوریم.

ابتدا من به کمک کتابخانه Pandas فایل csv موجود را خواندم و به کمک فانکشن **head** مدل یک دید تقریبی از داده‌های موجود در data set ام بدست آورم.

```
import pandas as pd
from numpy import mean

df = pd.read_csv('Lateraliry.csv')
df.head()
```

	Number	seizure_Simiology_Lateraliry	Lateraliry_ictal_EEG	ictal_Epileptogenic_zone	interictal_irritative_zone	MRI_findings
0	1	L	L	L	UL	L
1	2	L	L	L	UR	L
2	3	U	UL	L	U	UR
3	4	U	UL	L	UL	L
4	5	U	R	R	R	R

و به کمک تابع **describe** می‌توانیم به صورتی پراکندگی داده‌ها را متوجه بشیم.

```
df.describe()
```

	Number	Hipp_Vol_LI	Hipp_FLAIR_LI	Cg_LI	Fx_LI	Hipp_MD_LI	Overall_Lateraliry_NO
count	35.000000	35.000000	35.000000	35.000000	35.000000	35.000000	35.000000
mean	17.742857	-0.016640	-0.097719	-0.032931	0.013771	-0.000054	0.371429
std	10.062577	0.061620	0.267247	0.046677	0.033438	0.000189	0.490241
min	1.000000	-0.129227	-0.558875	-0.140300	-0.046900	-0.000765	0.000000
25%	9.500000	-0.063088	-0.325745	-0.060300	-0.013550	-0.000141	0.000000
50%	18.000000	-0.017164	-0.073292	-0.031900	0.020600	-0.000024	0.000000
75%	25.500000	0.007082	0.079581	-0.001750	0.042000	0.000042	1.000000
max	36.000000	0.106508	0.499853	0.080900	0.071600	0.000284	1.000000

با بررسی تصاویر بالا متوجه شدم که تعداد ۲۱ سطر ستون دارم که بیشتر از ۳۶ عدد از داده‌ها را بیشتر در اختیار ندارم و داده‌ها به صورت categorical و عددی وجود دارند. عددها نرمالایز شده هستند و داده‌های categorical را باید به کمک one hot encoding داشته باشیم.

در قدم بعد اسم ستون‌ها را چاپ می‌کنیم و label ها را از داده‌ها جدا می‌کنیم.

```
print(df.columns)
label = df['Overall_Laterality_NO']
del df['Overall_Laterality_NO']

Index(['Number', 'seizure_Simiology_Laterality', 'Laterality_ictal_EEG',
       'ictal_Epileptogenic_zone', 'interictal_irritative_zone',
       'MRI_findings', 'Hipp_Vol_LI', 'hippocampal volume', 'Hipp_FLAIR_LI',
       'FLAIR signal intensity', 'Cg_LI', 'FA cingulum', 'Fx_LI', 'FA fornix ',
       'Hipp_MD_LI', 'hippMD', 'Logistic prediction',
       'Side_ C4.5 Decision Tree', 'Handedness', 'gender',
       'Overall_Laterality_NO'],
      dtype='object')
```

سپس ابتدا به هر کدام از داده‌های categorical یک عدد انتصاب می‌دهیم.

۲ به معنای چپ ۱ به معنای نسبتاً چپ و صفر به معنای وسط و ۱- به معنای نسبتاً راست و ۲- به معنای راست
تقسیم بندی می‌کنیم و همچنین به مرد عدد ۱ و زن عدد ۱- و both عدد صفر را اطلاق می‌دهیم.

```
df = df.replace(to_replace=["L"],value=int(2))
df = df.replace(to_replace=["UL"],value=int(1))
df = df.replace(to_replace=["R"],value=-int(-2))
df = df.replace(to_replace=["UR"],value=int(-1))
df = df.replace(to_replace=["U"],value=int(0))
df = df.replace(to_replace=["female"],value=int(-1))
df = df.replace(to_replace=["male"],value=int(1))
df = df.replace(to_replace=["Both"],value=int(0))
df.head()
```

Number	seizure_Simiology_Laterality	Laterality_ictal_EEG	ictal_Epileptogenic_zone	interictal_irritative_zone	MRI_findings
0	1	2	2	2	1
1	2	2	2	-1	2
2	3	0	1	2	0
3	4	0	1	2	1
4	5	0	2	2	2

سپس برای one hot code کردن به ازای هر داده categorical یک ستون در نظر می گیریم و در صورت دارا بودن آن ویژگی توسط آن ستون مقادیر صفر و یک را مقدار دهی می کنیم .

```
list_col_new = ['Number']
list_col = ['seizure_Simiology_Laterality', 'Laterality_ictal_EEG',
            'ictal_Epileptogenic_zone', 'interictal_irritative_zone',
            'MRI_findings', 'hippocampal volume', 'FLAIR signal intensity',
            'FA cingulum', 'FA fornix ',
            'hippMD' , 'Logistic prediction', 'Side_ C4.5 Decision Tree', 'Handedness']

# make one hot coding for categorical feature
for name in list_col:
    df[f"L_{name}"] = df[f"{name}"] == 2
    list_col_new.append(f"L_{name}")

    df[f"UL_{name}"] = df[f"{name}"] == 1
    list_col_new.append(f"UL_{name}")

    df[f"R_{name}"] = df[f"{name}"] == -2
    list_col_new.append(f"R_{name}")

    df[f"UR_{name}"] = df[f"{name}"] == -1
    list_col_new.append(f"UR_{name}")

    df[f"U_{name}"] = df[f"{name}"] == 0
    list_col_new.append(f"U_{name}")

    del df[f'{name}']

df["female"] = df["gender"] == -1
df["male"] = df["gender"] == +1
```

سپس ستون Number را حذف می کنیم که index هر کدام از داده ها را مشخص می کند .

سپس یک مدل AdaBoost با ۵۰۰ عدد درخت تصمیم ساختم .

یک فرا پارامتر مهم برای الگوریتم AdaBoost تعداد درخت های تصمیم مورد استفاده در مجموعه است.

هر درخت تصمیمی که در گروه استفاده می شود، به گونه ای طراحی شده است که یک Weak learner باشد.

یعنی بر پیش بینی تصادفی مهارت دارد، اما مهارت بالایی ندارد. به این ترتیب، از درخت های تصمیم یک سطحی استفاده می شود که به آن decision stumps نیز می گویند.

تعداد درخت های اضافه شده به مدل باید زیاد باشد تا مدل به خوبی کار کند، اغلب صدها، اگر نه هزاران.

تعداد درخت ها را می توان از طریق آرگومان "n estimates" تنظیم کرد و به طور پیش فرض 50 را تنظیم کرد.

سپس به کمک KFold مشخص می کنیم که cross val score ما دیتاست را به چند بخش تقسیم کند .

و سپس الگوریتم cross val را بر روی آن انجام می دهیم .

در قسمت scoring می توانیم مشخص کنیم که scoring براساس کدام یک از معیارهای اندازه گیری خطا یعنی accuracy یا recall یا F1 بر روی داده ها train را انجام دهد و در نهایت مقدار دقت آن به چه مقداری می رسد نسبت به آن معیار .

سپس یک لیست از اعداد به ازای هر کدام از fold ها داریم که میانگین گرفتن از آن می توانیم میانگین F1 یا accuracy یا recall بدست آوریم .

```
from sklearn.ensemble import AdaBoostClassifier
from sklearn.model_selection import KFold, cross_val_score

del df['Number']
for column in df.columns:
    if df[column].isnull().values.any():
        print(column)
X = df.values.tolist()
clf = AdaBoostClassifier(n_estimators=500, random_state=0)
cv = KFold(n_splits=5, random_state=1, shuffle=True)
accuracy = cross_val_score(clf, X, label, scoring='accuracy', cv=cv, n_jobs=-1)
```

حال دقت بدست آمده از یک از K ها را براساس موارد یاد شده در بالا بررسی میکنیم .

k	recall	F1	accuracy
5	0.73	0.67	0.7442
7	0.5952	0.6	0.74
10	0.5166	0.446	0.75

یکی از معیارهایی که باید در بررسی مدل مدنظر قرار بدهیم این است که برای ما کدام شاخص مهتر است .
از آنجا که در مدل های مبتنی بر تشخیص بیماری برای ما مهتر است که FN (false negative) در کمترین مقدار خودش باشد یعنی به اشتباه نگویم بیماری بیمار نیست پس مدل با $k = 5$ بهتر است برای ما نسبت به مدل با $k = 10$ زیرا عملا در مدل با $k = 10$ پیدا کردن گونه هایی که بیمار نیستند به شکلی شانی در حال بررسی هست و دقت بسیار بدی دارد .

و همانطور که می بینید $F1$ $k = 5$ نیز همه بهتر است .

Covid-19:

برای سوال دوم که از ما خواسته شده براساس مجموعه دادگان استخراج شده از تصاویر CT بیماران استفاده کنیم و تفاوت بین دو مدل AdaBoost و Tree Decision را بر روی مدل بررسی کنیم .
ابتدا مثل سوال قبل به بررسی داده می پردازیم .

```
import pandas as pd

df = pd.read_csv('COVID_Dataset.csv')
df.head()
```

	Unnamed: 0	24	25	26	27	28	29	30	31	32	...
0	0	12.649111	12.649111	14.866869	14.866869	14.866869	17.916473	1315.166667	12.649111	0.845494	...
1	1	12.649111	12.649111	14.866869	14.866869	14.866869	17.916473	1315.166667	12.649111	0.845494	...
2	2	12.649111	12.649111	14.866869	14.866869	14.866869	17.916473	1315.166667	12.649111	0.845494	...
3	3	12.649111	12.649111	14.866869	14.866869	14.866869	17.916473	1315.166667	12.649111	0.845494	...
4	4	12.649111	12.649111	14.866869	14.866869	14.866869	17.916473	1315.166667	12.649111	0.845494	...

همانطور که می بینیم که یک سری از داده ها به صورت تکراری در تمامی سطرها تکرار شده اند .

سپس با توجه به دستور زیر سعی می کنیم این ستون ها که در تمامی سطرها یکسان هستند را حذف کنیم .

```
for column in df.columns:
    if (df[column] == df[column][0]).all():
        #f (df[column].isnull().values.all()):
            del df[column]

df.head()
```

	36	37	38	39	40	41	42	43	44	45	...
0	-0.134780	-0.024410	14.340823	0.343471	0.032337	22.312203	0.430488	0.040340	-0.081470	-0.074048	...
1	-0.085899	0.054907	33.993878	0.672491	0.047374	21.790254	1.173460	0.079642	-0.007310	-0.046421	...
2	0.575542	1.048788	1218.670868	0.339670	0.044061	10.261242	1.105904	0.216354	0.886834	1.009623	...
3	0.899426	1.715812	1997.735245	0.052993	0.475470	3.851524	2.129827	0.285866	1.169449	1.045625	...
4	-0.516361	-0.092693	240.693081	0.356972	0.151774	8.379730	0.673481	0.135732	-0.378181	-0.454677	...

سپس به کمک دستور زیر داده های ستون را normaliz میکنیم به سبک **Data Standardization** .

- $y = (x - \text{mean}) / \text{standard_deviation}$

```
list_feature = []
for column in df.columns:
    if column != "Infection":
        list_feature.append(column)
        df[column] = (df[column] - df[column].mean()) / df[column].std()

df.head()
```

	36	37	38	39	40	41	42	43	44	45	...
0	-0.146209	-0.891035	-0.926751	-0.195508	-0.769381	1.587472	-0.900329	-0.907143	-0.547320	-0.479426	...
1	-0.067955	-0.776731	-0.901499	0.698730	-0.783803	1.508466	-0.058343	-0.697600	-0.424162	-0.404343	...
2	0.990954	0.655559	0.636706	-0.211425	-0.793021	0.246908	-0.137679	0.031294	1.060653	1.260471	...
3	1.509464	1.616814	1.648257	-0.995395	0.407331	-0.454473	1.064779	0.401905	1.529962	1.317228	...
4	-0.757087	-0.989438	-0.633118	-0.164111	-0.493321	0.041024	-0.645500	-0.398549	-1.040029	-1.047944	...

سپس به کمک model_selection کتابخانه skitlearn داده train , test همراه با label های آن ها را با

فرض اینکه train size به میزان ۷۵ درصد کل داده باشد و ۲۵ درصد تقسیم بندی کردیم با

random state به مقدار ۱۲ .

```
from sklearn.model_selection import train_test_split

X = df.loc[:, list_feature]
y = df.loc[:, ['Infection']]
X_train, X_test, y_train, y_test = train_test_split(X, y, random_state=12, train_size = .75)

X_train.head()
```

<	<	5 rows	>	>	5 rows × 93 columns				
÷	36 ÷	37 ÷	38 ÷	39 ÷	40 ÷	41 ÷	42 ÷	43 ÷	44 ÷
48068	-0.143493	-0.061686	-0.800661	1.369552	-0.273492	-0.354402	-0.241290	0.029183	-0.302705
15951	0.655297	1.935581	1.948211	-0.726376	1.916300	-0.660127	1.272566	1.593118	1.499993
19502	-0.792597	-1.376062	-0.546261	-1.095851	-0.738841	0.281746	-1.429878	-0.813179	-1.197228
25532	-0.793483	-1.233658	-0.559284	-0.479985	-0.731939	0.257078	-1.063581	-0.599183	-1.153054
67302	1.022416	1.022800	0.057675	1.116926	0.162160	0.530851	0.705810	0.112767	1.011707

سپس مدل Adaboost با تعداد درخت تصمیم به تعداد ۱۰۰۰ عدد و برای بدست آوردن ، accuracy ، precision ، recall و roc_auc از کتابخانه metrics از sklearn استفاده می کنیم براساس مدل fit شده بر روی داده train و بر روی داده تست دقت هر کدام را بدست می آوریم .

```

from sklearn.ensemble import AdaBoostClassifier

import numpy as np

clf = AdaBoostClassifier(n_estimators=1000, random_state=1)
x_t = np.array(X_train.values.tolist())
y_t = np.array(y_train.values.tolist()).reshape(len(y_train))
clf.fit(x_t, y_t)

```



AdaBoostClassifier

AdaBoostClassifier(n_estimators=1000, random_state=1)

```

x_te = np.array(X_test.values.tolist())
y_te = np.array(y_test.values.tolist()).reshape(len(y_test))

```

```

from sklearn import metrics

```

```

y_pred = clf.predict(x_te)
print("Accuracy:", metrics.accuracy_score(y_te, y_pred))

```

Accuracy: 0.7776

```

print("F1:", metrics.f1_score(y_te, y_pred))

```

F1: 0.6358277386605534

سپس به کمک مدل tree از sklearn یک درخت تصمیم را بر روی داده‌های train ست می‌کنیم و سپس از آن برای بدست آوردن کارایی مدل براساس دقت های گفته شده در بالا و برروی داده تست استفاده می‌کنیم و نتیجه را گزارش می‌کنیم .


```
from sklearn import tree
tree = tree.DecisionTreeClassifier()
tree.fit(x_t,y_t)
```

▼ DecisionTreeClassifier
DecisionTreeClassifier()

```
tree_y_pred = tree.predict(x_te)
```

```
print("Accuracy:",metrics.accuracy_score(y_te, tree_y_pred))
```

```
Accuracy: 0.73745
```

```
print("F1:",metrics.f1_score(y_te, tree_y_pred))
```


```
F1: 0.6138118702654998
```

```
print("Recall:",metrics.recall_score(y_te, tree_y_pred))
```

```
Recall: 0.6284638554216867
```

مدل	Accuracy	F1	Recall	Precision	Roc Auc
Decision Tree	0.73745	0.6138	0.62846	0.5998	0.71
Adaboost	0.7776	0.6358	0.584	0.6966	0.73

همانطور که مشاهده می کنید در تمامی موارد به غیر از Recall مدل Adaboost بهتر از Decision Tree کار می کند ولی از آنجا که این اختلاف فقد چهار درصد است می توان امید داشت با افزایش پرامترهای درخت تصمیم یا عمق درخت تصمیم Adaboost به نتایج بهتری رسید . مثلاً با ۲۰۰۰ تا به نتیجه زیر می رسم .

مدل	Accuracy	F1	Recall	Precision	Roc Auc 
Decision Tree	0.73745	0.6138	0.62846	0.5998	0.71
Adaboost	0.78	0.64	0.596	0.6980	0.734

پس همانطور که می بیند میشه recall را با موارد دیگه کم و کمتر کرد در adaboost ما که در این قسمت هم نتیجه ای بهتر از decision Tree به ما برگردوند .