دانشگاه صنعتی امیرکبیر
(پلی تکنیک تهران)

دانشکده مدیریت، علم و فناوری

پروژه ۵ درس داده کاوی

بررسی روش‌های طبقه بندی مختلف بر روی دیتاستی مربوط به پارامترهای رانندگی و طبقه بندی کردن راننده‌های مختلف

نگارش
فاطمه سادات علیخانی

استاد درس
دکتر مهدی قطعی

فهرست

مقدمه :

طبقه بندی (classification) یکی از روش های یادگیری ماشین است و برای یادگیری چگونگی تخصیص برچسب کلاس به یک نمونه ورودی، استفاده می شود. به این صورت که ابتدا با استفاده از داده‌های train یک مدل را آموزش میدهیم و سپس با استفاده از آن لیبل نمونه های جدید را مشخص می‌کنیم در این گزارش سعی شده با استفاده از الگوریتم های decision treeو k- nearest neighborو svm مدل را آموزش داده و سپس سعی میکنیم با بررسی هایپر پارامترهای مختلف بهترین مدل را انتخاب کنیم و بعد از مدل را بر روی داده‌های Test می‌سنجیم. این الگوریتم‌ها بر روی یک دیتاست مربوط به پارامترهای رانندگی بررسی شده است و حالات رانندگی مختلف را به وسیله آنها دسته بندی می‌کنیم.

معرفی دیتاست

در این گزارش از دیتاست Driving behavior Dataset در سایت mendeley data استفاده شده است و حاوی ۱۱۱۴ سطر و ۷ ستون است. که ستون‌های آن شامل شتاب ماشین در سه محور x, y, z (Accx, Accy , Accz) است و جهت‌گیری و سرعت زاویه‌ایی ماشین که میتواند چرخش ماشین را تشخیص دهد حول سه محور x, y, z است.(Gyrox, Gyroy, Gyroz). به همراه یک ستون(class) که در آن یکی از چهار حالت زیر را نشان می‌دهد:

۱. شتاب ماشین به صورت ناگهانی افزایش پیدا کرده و سرعت آن زیاد می‌شود.(Sudden Acceleration)

۲. به سمت چپ چرخش ناگهانی دارد.(Sudden left Turn)

۳. به سمت راست چرخش ناگهانی دارد. (Sudden Right Turn)

۴. یا به صورت ناگهانی شتاب آن کاهش پیدا میکند و ماشین به مانع برخورد میکند.(sudden break)

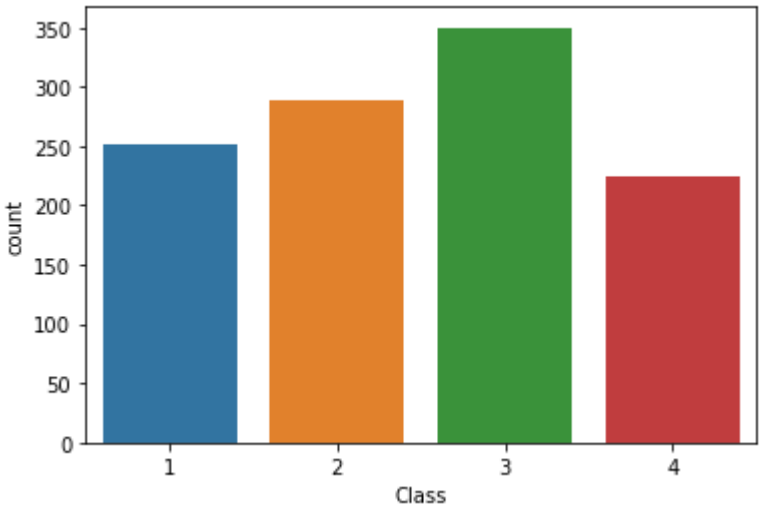
این اطلاعات در شرایطی جمع آوری شده که وضعیت بارندگی نبوده و سطح جاده خشک بوده است. همینطور حسگرها برای سنجش شتاب و سرعت زاویه ایی بر روی داشبورد ماشین قرار داشته و توسط ۳ راننده جمع آوری شده است. قسمتی از داده‌ها به شکل زیر است :

	Class	GyroX	GyroY	GyroZ	AccX	AccY	AccZ
0	1	-0.923664	3.694656	0.824427	0.162598	-0.086670	-0.969482
1	1	-0.908397	4.534351	0.832061	0.175781	-0.100586	-1.013184
2	1	0.786260	3.969466	0.587786	0.322754	-0.140381	-0.911621
3	1	0.335878	4.564885	-0.251908	0.480225	-0.226807	-0.936768
4	1	3.351145	2.694656	-0.106870	0.426025	-0.253906	-0.950195

شکل ۱- بخشی از دیتاست

مصور سازی

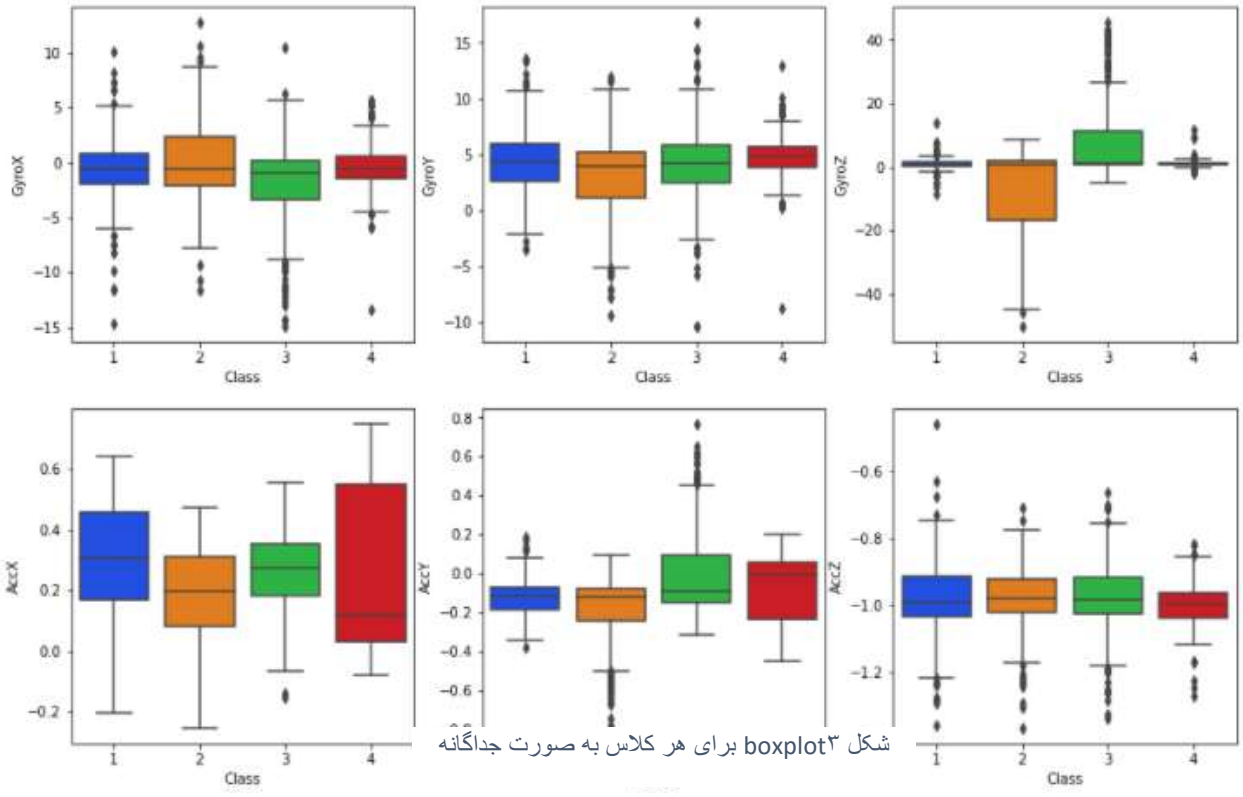
برای آشنا شدن بیشتر با دیتاست و درک بهتر آن سراغ مصور سازی میرویم شکل۲ نشان میدهد که هر یک از چهار کلاس گفته شده چه تعداد نمونه داریم.



شکل ۲- countplot برای نمونه های هر کلاس

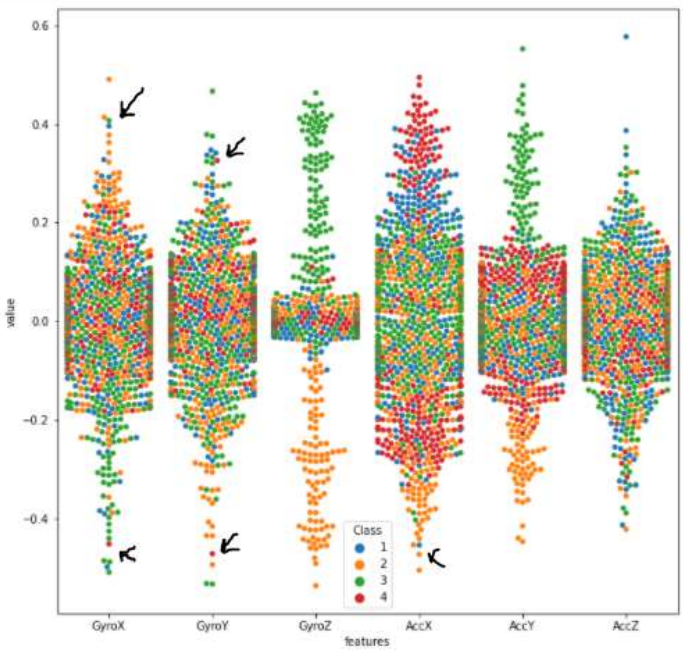
همانطور که در شکل نشان داده میشود تعداد نمونه ها در کلاس ۳ ینی گردش ناگهانی به چپ بیشتر از بقیه کلاس هاست و از کلاس توقت ناگهانی و برخود ماشین به سطح تعداد نمونه کمتری داریم.

شکل ۳ نمودار باکس پلات هر کلاس را به تفکیک نشان میدهد و در هر کدام به تفکیک داده‌های outlier آن مشخص است مثلا برای ستون accx فقط کلاس ۳ دارای داده پرت است.



شکل ۳boxplot برای هر کلاس به صورت جداگانه

برای بهتر دیدن داده ها از شکل ۴ استفاده میکنیم که در آن نمونه های مربوط به هر کلاس را به تفکیک نشان میدهد. نمونه هایی که با فلش مشخص شده اند باعث ایجاد مشکل در مراحل بعدی و طبقه بندی ما میشود پس در مرحله پیش پردازش داده یک سری داده های outlier مربوط به قسمت های مشخص شده را با میانگین داده های همان کلاس در ستون جایگزین میکنیم تا در طبقه بندی ما دچار مشکل نشود.



شکل ٤

(همینطور در شکل ۳ برای ستون Gyroz کلاس های ۲ و ۳ دارای داده پرت زیادی هستند اما آنها را برای جایگزین کردن مقادیرشان با میانگین کلاس مربوطه انتخاب نکردم به این دلیل که داده ها خود به صورت جدا شده هستند و در طبقه بندی ایجاد مشکل نمیکنند.)

پیش پردازش داده‌ها

a. جایگزینی برخی از outlier ها

همانطور که در قسمت قبل گفته شد برخی از داده های پرت در طبقه بندی ما باعث ایجاد مشکل می‌کنند مثلا با توجه به شکل ۴ کلاس۳(سبز) فقط یک نمونه از آن در ستون ACCX کمتر از ۰,۴- دارد و ... این داده ها را با میانگین نمونه های دیگر کلاس ۳ در ستون Accx جایگزین می‌کنیم بخشی از خروجی کد به شکل زیر است:

```
3 AccX
Q1 , Q2 , Q3 , IQR :  0.18359375 0.27465820350000003 0.3524169925 0.16882324250000003
outlier data : [-0.136230469, -0.150878906]
```

(نکته : همه ی داده های پرت را به این روش جایگزین نمی‌کنیم، فقط آنهایی که با توجه به شکل ۴ باعث ایجاد مشکل در طبقه بندی شده اند را جایگزین کرده.)

b. نرمال سازی داده ها

نرمال سازی داده ها باعث میشود که همه‌ی داده ها در یک رنج قرار بگیرند و بتوان ستون های مختلف را با هم مقایسه کرد و StandardScaler در کتابخانه sklearn برای نرمال‌سازی همه‌ی ستون ها به جز ستون class استفاده شده است و قسمتی از داده ها به صورت زیر است :

```
array([[-0.02753838, -0.13153147, -0.02092141, -0.49062104,  0.04434692,
         0.14009952],
       [-0.02305169,  0.12673042, -0.0202876 , -0.41860806, -0.02877161,
        -0.3057532 ],
       [ 0.47497117, -0.0470094 , -0.04056951,  0.38420329, -0.23786493,
         0.73041848],
```

تقسیم داده ها به داده هایTrain و Test

برای اجرای الگوریتم های طبقه بندی نیاز است که داده ها را به سه دسته‌ی train, test validation دسته بندی کرد داده‌های train را برای آموزش مدل استفاده کنیم و داده‌ی validation را برای ارزیابی مدل استفاده میکنیم، و در انتها مدل را با داده تست می‌سنجیم.

برای این کار از متد train_test_split در کتابخانه sklearn استفاده شده است و دیتاست را به نسبت ۷۰ – ۳۰ تقسیم بندی کرده . تعداد داده های آموزش و تست به زیر است :

```
Train set: (779, 6) (779,)
Test set: (335, 6) (335,)
```

با توجه به قسمت مصور سازی در شکل۲ نشان داده شد که تعداد نمونه‌ها برای کلاس های مختلف متفاوت است و ممکن است که داده های train برای یک کلاس کمتر شود و مدل برای آن کلاس کمتر آموزش ببیند. برای حل این مشکل از متغیر stratify استفاده می‌کنیم که در آن داده های train به همان نسبتی که در دیتاست اصلی داریم تقسیم میشود.

پیاده سازی الگوریتم های Classification

الگوریتم kkn

توضیح الگوریتم

در این الگوریتم برای هر مشخص کردن هر نمونه جدید باید k نزدیک ترین همسایه نمونه را پیدا کرد و با توجه به کلاس این همسایه ها کلاس نمونه جدیدمان را بدست آوریم. پس در این الگوریتم چند پارامتر مورد بررسی داریم ۱. مقدار k ۲. فاصله را چطور حساب کنیم ۳. ارزش همسایه ها را چطور حساب کنیم

برای پیاده سازی از KNeighborsClassifier در کتابخانه sklearn استفاده شده است.

- متغیر metric برای این است که فاصله بین نمونه ها را به چه صورت محاسبه کند. به طور مثال میتوان فاصله هر دو نمونه را به صورت فاصله اقلیدسی یا منهتن حساب کرد.

-متغیر weight دارد به معنی این که ارزش هر همسایه را چطور محاسبه کنیم به طور مثال اگر uniform باشد ارزش همه ی همسایه ها برابر است اما اگر distance باشد به نسبت عکس فاصله ها متغیرها ارزش دارند ینی هر چه یک نمونه فاصله کمتری با نمونه‌ی مورد بررسی ما داشته باشد ارزش آن برای دسته بندی بیشتر می‌شود.

بدست آوردن بهترین مدل

برای بررسی اینکه چه مدلی روی داده ها از بقیه بهتر عمل میکند از k-fold cross validation استفاده خواهیم کرد.

به این صورت که مثلا دیتای آموزش را به ۱۰ قسمت تقسیم کرده و هر دفعه یکی از این قسمت ها را به عنوان داده‌ی validation برای ارزیابی مدل استفاده میکنیم و دقت آن را با میانگین گرفتن دقت هر ۱۰ مرحله بدست می آوریم که همه‌ی داده های train در محاسبه دقت نقش داشته باشند.

این عملیات را استفاده از پارامتر های مختلف انجام می‌دهیم و بهترین دقت را انتخاب کرده و آن مدل را انتخاب میکنیم.در این جا پارامتر های زیر با هم بررسی شده اند:

```
ks = [1,3,5,7,8,9,10,11,12,13,14,15,16,17,18]
weights =  ['uniform','distance']
metrics =  ['euclidean','manhattan']
hyperprameter_candidates = [{'n_neighbors': ks, 'weights': weights , 'metric' : metrics}]
```

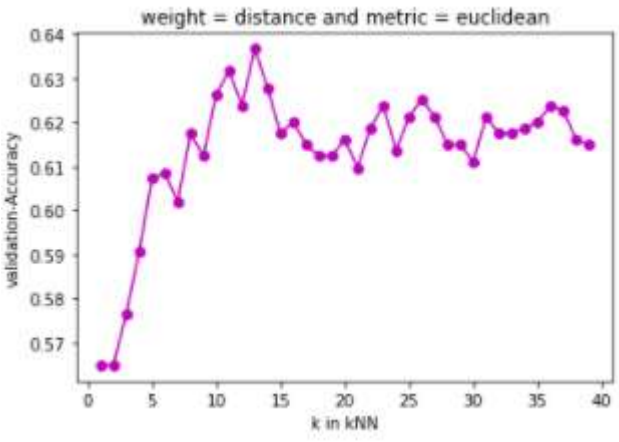
حال ترکیبی از اینها را مقایسه کرده (به طور مثال k=8 , weight = ‘uniform’ , metric = ‘Manhattan’) و همینطور میانگین دقت validation را برای آن حساب می‌کنیم و بهترین دقت validation را به عنوان بهترین مدل معرفی میکنیم.

این کار را با تابع GridSearchCV در کتابخانه sklearn انجام می‌دهیم و نتیجه آن به صورت زیر است :

```
best accuracy : 0.6316350316350315
best prameter : {'metric': 'euclidean', 'n_neighbors': 11, 'weights': 'distance'}
```

بهترین دقت برای داده های validation ۰,۶۳ است که با k=11 و weight = distance و Metric = Euclidean محاسبه می‌شود.

شکل زیر نشان دهنده دقت‌های مختلف برای الگوریتم knn با k های ۱ تا ۴۰ و Metric = Euclidean و weight = distance است. همانطور که در شکل مشخص است دقت داده های validation تا عدد ۱۱ زیاد میشود و بعد از آن کاهش میابد.



پس k=11 میتواند بهترین مقدار برای این مدل باشد.

همینطور فاصله اقلیدسی مقدار مناسب ترین نسبت به فاصله منهتن محاسبه میکند.

و ارزش هر نمونه ی همسایه طبق نزدیکی فاصله آن با نمونه مورد بررسی محاسبه میشود پس میتواند مقدار دقیق تری را برای ما محاسبه کند.

اما زمانی که دقت داده ی train را محاسبه میکنیم این مقدار برابر یک میشود

```
print("Train set Accuracy: ", metrics.accuracy_score(y_train, best_knn.predict(X_train)))
```

```
Train set Accuracy:  1.0
```

که این به معناست که مدل overfit شده و بیش از حد داده ها را حفظ کرده است، این به این دلیل است که پیچیدگی مدل اضاف شده است مثلا استفاده از Weight = distance باعث این اتفاق شده است.

پس دوباره الگوریتم gridSearchCV را اجرا میکنیم و مقدار weight را برابر uniform قرار میدهیم تا از پیچیدگی مدل کم شود. پارامتر های بررسی شده به صورت زیر است :

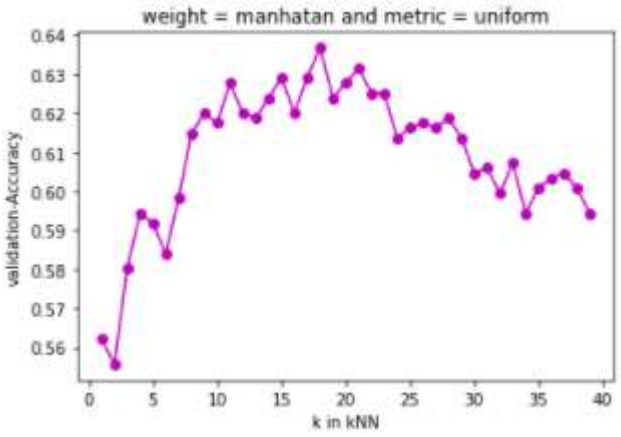
```
ks = [1,3,5,7,8,9,10,11,12,13,14,15,16,17,18,19,20]
weights =  ['uniform']
metrics =  ['euclidean','manhattan']
hyperprameter_candidates = [{'n_neighbors': ks, 'weights': weights , 'metric' : metrics}]
```

و نتیجه بدست آمده به عنوان بهترین مدل به صورت زیر است :

```
best accuracy : 0.6368131868131868
best prameter : {'metric': 'manhattan', 'n_neighbors': 18, 'weights': 'uniform'}
```

با توجه به جواب بدست آمده معیار بدست آوردن فاصله manhattan است و مقدار k برابر ۱۸ بدست آمده است . شکل زیر نشان دهنده دقت داده های Validation برای الگوریتم knn بین k=1 تا k=40 است.

همینطور metric = manhattan و weight = uniform است.



همانطور که در شکل رو به رو مشاهده می‌شود بیشترین مقدار دقت در داده های بین ۱۵ تا ۲۰ است .

و دقت داده اموزش به صورت زیر است :

```
print("Train set Accuracy: ", metrics.accuracy_score(y_train, best_knn.predict(X_train)))
```

```
Train set Accuracy:  0.6829268292682927
```

بررسی داده تست

پس از به دست آوردن مدل نهایی دقت آن را بر روی داده ی تست بررسی میکنیم که این دقت به صورت زیر است :

```
Test set Accuracy:  0.582089552238806
Train set Accuracy:  0.6829268292682927
```

همینطور میتوان اطلاعات مدل را به طور کلی بر روی داده های تست بررسی کرد که به صورت زیر است :

همینطور که در این شکل نمایش داده میشود

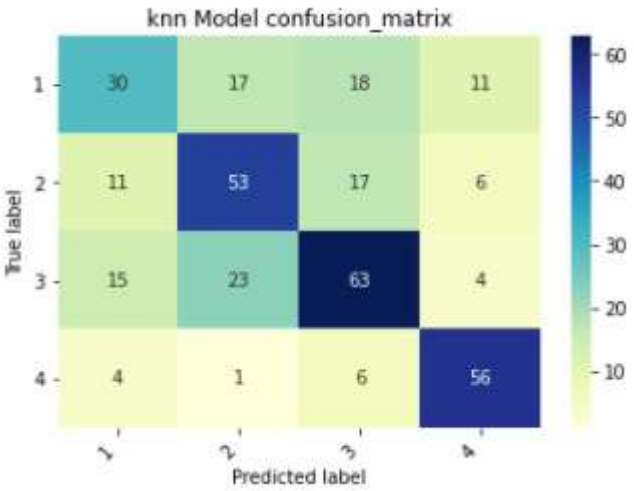
	precision	recall	f1-score	support
1	0.50	0.39	0.44	76
2	0.56	0.61	0.59	87
3	0.61	0.60	0.60	105
4	0.73	0.84	0.78	67
accuracy			0.60	335
macro avg	0.60	0.61	0.60	335
weighted avg	0.60	0.60	0.60	335

Precision نشان دهنده مقداری است که از بین کلیه مقادیری که به طور مثال به عنوان کلاس ۱ سنجیده شده اند چه تعداد درست بوده است. این مقدار برای کلاس ۴ که برخورد کردن به مانع و ایستادن خورد است بیشترین است (ینی کلاس چهارم به نسبت بقیه کلاس ها بهتر سنجیده شده است.)

مقدار recall نشان دهنده این است که از نمونه هایی که برچسب کلاس یک را داشند چه تعداد به درستی سنجیده شده اند که همانطور که در شکل میبینیم کلاس یک که به معنای بالا رفتن سرعت ناگهانی ماشین بود دقت پیشبینی پایین تری به نسبت بقیه دارد و کلاس ۴ را بهتر از بقیه کلاس ها میتوان تشخیص داد.

مقدار f1-score برابر 2*(prc*recall)/(pre+recall) نشان دهنده ی مقدار دقت کلی برای پیش بینی هر کلاس است که برای کلاس ۴ بیشترین دقت و برای کلاس ۱ کمترین دقت را داریم.

همینطور در شکل روبه رو میتوان مقادیر پیش بینی شده را بررسی کرد.



- مقادیری از تست که در کلاس چهار بودند و پیشبینی آنها نیز کلاس چهار بوده برابر۵۶ داده بوده .
- آنهایی که در اصل در کلاس ۴ بوده اند و به عنوان کلاس ۳ شناخته شده اند ۶ تا بوده اند که این موضوع میتواند مهم باشد که برخورد به مانع را چرخش به راست تشخیص دادیم و ممکن است باعث ضرر کردن سیستم شود.

و به همین ترتیب برای بقیه ی کلاس ها نیز بررسی میشوند.

الگوریتم decision Tree

توضیح الگوریتم

حالات مختلف بر روی یک درخت را حساب کرده و می‌سنجد که طبق چه حالتی میزان entropy کمتر و information gain بیشتر داریم و به همین ترتیب شاخه های درخت را محاسبه میکند تا مدل مورد نظر ساخته شود.

برای پیاده سازی این الگوریتم از تابع DecisionTreeClassifier در کتابخانه sklearn استفاده می‌کنیم.

پارامتر هایی که این متد به عنوان ورودی میگیرد یکی این است که entropy را برای بدست آوردن information gain استفاده کند یا متد gini و اینکه تا چه عمقی از درخت محاسبه الگوریتم را ادامه دهد و همینطور هر دفعه محاسبه چه تعداد از ستون ها را چک کند و information gain آن را بدست آورد.

$$Gini = 1 - \sum_{i=1}^{n}(p_i)^2 \quad E(S) = \sum_{i=1}^{c} -p_i log_2 p_i$$

همینطور متغیر هایی نظیر اینکه min_samples_split که نشان دهنده این است که هر نود چه تعداد سمپل داشته باشد تا آن را ادامه دهیم دارد.

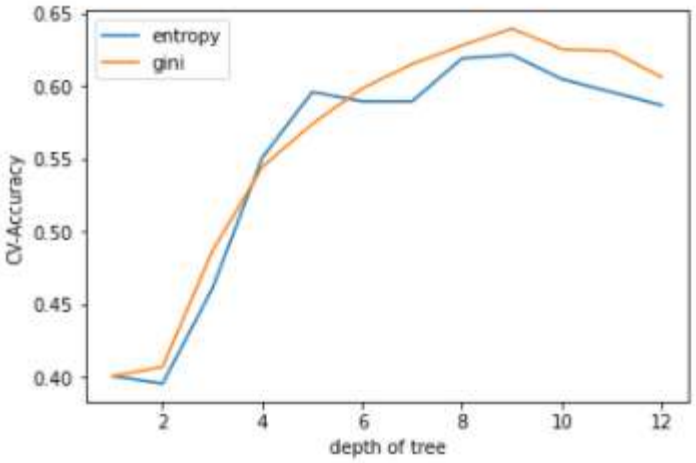
بدست آوردن بهترین مدل

این متغیر ها به صورت شکل رو به رو به تابع gridSearchCV که در قسمت قبل توضیح داده شد داده شده اند تا بهترین مدل با استفاده از این متغیر ها بدست بیاید .

```
criterion = ['gini', 'entropy']
max_depth = [6,7,8,9,10,11,12,13,14,15]
max_features = ['sqrt', 'log2' , 0.90]
```

و نتیجه آن به صورت شکل زیر است :

```
best score : 0.6277056277056278
best prams : {'criterion': 'gini', 'max_depth': 9, 'max_features': 0.9}
```

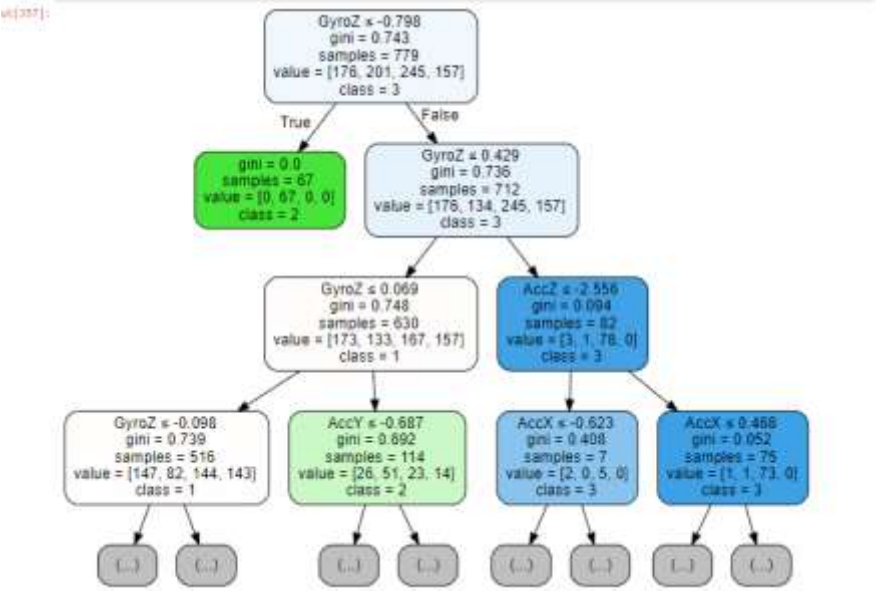


بهترین مقدار ارتفاع برابر ۹ است و ۰,۹ تعداد کل ستون ها بست داده شده است و متد برای بدست آوردن information gain ، gini است.

دقت داده های validation برای متد gini و entropy را در مدل هایی با عمق های مختلف درخت محاسبه کرده و شکل آن به صورت روبه رو است

همانطور که در شکل مشخص است دقت اندازه گیری این دو متد بسیار به هم نزدیک هستند و در مقدار depth=9 بیشترین دقت را داراست و پس از آن دقت مدل کم میشود.

قسمتی از درخت تشکیل شده به صورت رو به رو است:



همانطور که مشاهده میشود در ریشه درخت مقدار gini برابر ۰,۷۴۳ است که با استفاده از gyroz>۰/۷۹۸ مقایسه میشود و در سمت چپ درخت یک که این مقدار gyroz<0.798 برقرار است با توجه به اینکه gini=0 است و اینکه در این دسته فقط کلاس دو مشاهده می‌شود و دسته ی کلاس دو به صورت کامل جدا میشود.

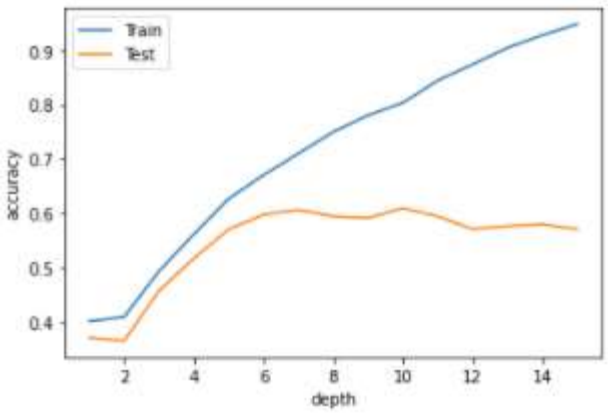
به همین ترتیب در سمت راست درخت ساخته میشود و این درخت ادامه پیدا میکند تا جایی که به max_depth برسیم یا تا جایی که یک نود خالص مانند نود ۲ ساخته شود.

بررسی داده تست

دقت داده های تست و train به صورت رو به رو است :

```
Test set Accuracy:  0.5834559378373881
Train set Accuracy:  0.7741886370632568
```

با توجه به اینکه دقت داده آموزش بسیار بیشتر از دقت داده تست است نمودار دقت اون دو دسته از داده را با عمق های درخت مختلف و متد gini بدست می آوریم و نمودار آن را رسم کرده و شکل زیر حاصل میشود :



همانطور که در شکل مشاهده میشود با افزایش عمق درخت

دقت داده های train بسیار زیاد شده و این به این دلیل است که پیچیدگی مدل بسیار زیاد شده و در آن فقط داده های ترین به خوبی شناخته شده و اگر مجموعه داده تست را به آن بدهیم دقت آن به خوبی داده های ترین نیست.

پس همانطور که گفته شد در داده های بیشتر از ۸ overfit رخ میدهد به این دلیل که دقت داده های train زیاد شده و دقت داده های تست تا مقدار ۸ زیاد و سپس کم میشود.

پس یکبار دیگر مدل را تغیر میدهیم و عمق های کمتر از ۸ را در آن بررسی می‌کنیم همینطور به جای اینکه همه‌ی ستون ها را چک کند یا لگاریتمی از آنها را بررسی میکند و یا جذر آنها را بررسی میکند این کار باعث میشود پیچیدگی مدل کمتر شود و overfit در آن کمتر اتفاق بیافتد.

```
criterion = ['gini', 'entropy']
max_depth = [4,5,6,7,8,9]
max_features = ['sqrt', 'log2' ]
```

و نتیجه تابع gridSearchCV که تمام این پارامتر ها را با هم مقایسه میکند و سپس بهترین ترکیب را برای مدل انتخاب میکند به صورت زیر است :

```
best score : 0.602147852147852
best prams : {'criterion': 'gini', 'max_depth': 7, 'max_features': 'log2'}
```

با توجه به این که max_depth=7 شد میدانیم در این حالت دیگر در حالت overfit نیست.

حال یکبار دیگر دیتای دقت دیتای train و test را محاسبه می‌کنیم و به صورت شکل زیر است

```
Test set Accuracy:  0.6013096560237443
Train set Accuracy:  0.7145619069063296
```

همینطور نتیجه طبقه بندی داده های تست به صورت زیر است :

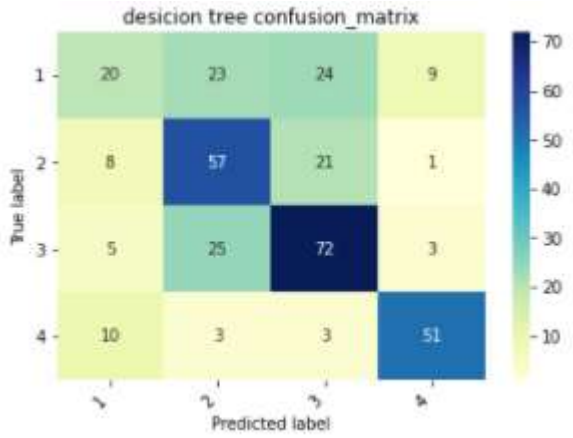
	precision	recall	f1-score	support
1	0.47	0.26	0.34	76
2	0.53	0.66	0.58	87
3	0.60	0.69	0.64	105
4	0.80	0.76	0.78	67
accuracy			0.60	335
macro avg	0.60	0.59	0.58	335
weighted avg	0.59	0.60	0.58	335

دقت نتیجه گیری در درخت تصمیم گیری کمی متفاوت نسبت به دقت در الگوریتم knn است.

مقدار recall برای کلاس ۱ بسیار پایین است به این معنی که داده هایی که در واقعیت لیبل کلاس آنها یک بوده است به درستی شناخته نشده اند. و از بین آنهایی که به عنوان کلاس یک شناخته شده اند فقط ۴۷ درصد آنها درست بوده(precision) که به این معنی ایست که با سرعت حرکت کردن راننده به خوبی تشخیص داده نشده است.

دقت پیشبینی کلاس ۴ بیشترین مقدار را داراست و اینکه ماشین به صورت ناگهانی توقف کند راحت از بقیه حالات قابل تشخیص است.

همینطور confusion matrix نتایج طبقه بندی شده به صورت شکل زیر است :



در این شکل مشاهده میشود (در ردیف اول) اکثر لیبل هایی که در اصل مقدار یک بوده اند. به عنوان کلاس ۲ یا ۳ طبقه بندی شده اند و این مقدار حتی بیشتر از مقداری است که برای کلاس یک اندازه گیری شده است.

همینطور کلاس ۲ و ۳ با اینکه به خوبی طبقه بندی شده اند اما گاهی به جای هم طبقه بندی شده اند به طور مثال ۲۱ نمونه که در کلاس ۲ بوده اند به عنوان کلاس ۳ طبقه بندی شده اند.

الگوریتم SVM

توضیح الگوریتم

الگوریتم دیگری که بررسی می‌کنیم الگوریتم svm است که در این الگوریتم سعی میشود بین کلاس‌های مختلف یک ابر صحفه رسم کند، هدف الگوریتم svm این است که بهترین مرز را برای جدا سازی کلاس‌ها پیدا کند. برای پیاده سازی این الگوریتم از متد SVM در کتابخانه sklearn استفاده می‌کنیم.

پارامترهایی که این الگوریتم دارد یکی kernel است که در اصل توابعی هستند که نوع رسم ابر صفحه و مشخص کردن مرز بین کلاسها را برای ما مشخص میکنند. کرنل هایی مانند linear, poly, rbf اگر کرنل به صورت خطی باشد مرز بین کلاسها هم به صورت خطی و در فضای دو بعدی مشخص میشود.

پارامتر بعدی Regularization یک پارامتر برای کنترل ارور است ، اگر این مقدار زیاد باشد به معنی ارور کم و اگر زیاد باشد به معنی ارور کم در مدل است، اگر این مقدار را بسیار زیاد باشد به نوعی میتوان گفت که مساله اورفیت خواهد شد زیرا سعی میکند طبقه بندی بدون هیچ اروری نسبت به داده آموزش اتفاق بیافتد و این باعث ایجاد مشکل میشود. و باید از مقدارهای زیاد آن اجتناب کرد.

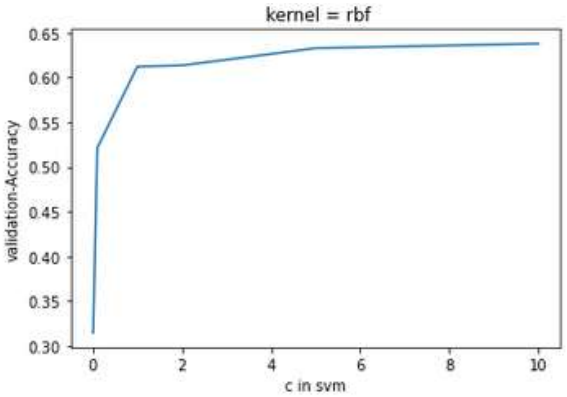
پارامترها به صورت رو به رو انتخاب شده اند.

و از متد GridSearchCV استفاده میشود که با توجه به مقدار دقت داده های validation بهترین مدلی که بهترین دقت را دارد انتخاب شود و نتیجه به شکل زیر است.

```
kernel =['linear', 'poly', 'rbf']
c = [0.001, 0.1, 1, 2]
hyperprameter_candidates = [{'kernel': kernel , 'C': c}]
```

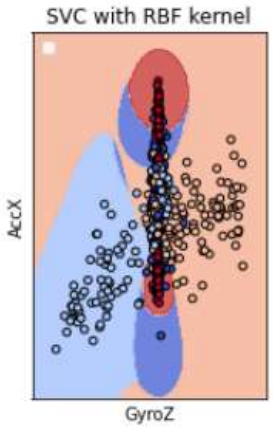
```
best accuracy : 0.6135364635364635
best prameter : {'C': 2, 'kernel': 'rbf'}
```

بیشترین و بهترین دقت برابر ۰,۶۲ بوده با کرنل rbf و مقدارc=2



شکل رو به رو نمودار دقت داده ها ارزیابی و cها مختلف با کرنل rbf را نشان می‌دهد. هر چه این مقدار بیشتر میشود دقت داده های validation نیز بیشتر می‌شود اما به معنی بهتر شدن مدل نیست چون باعث over fit آن می‌شود.

با استفاده از decision plot می‌شود میتوان در یک شکل دو بعدی مرزهای بین کلاس های دسته بندی شده را مشاهده کرد:



بررسی داده تست

دقت داده آموزش و تست به صورت رو به رو است :

```
Test set Accuracy:  0.5928922441203368
Train set Accuracy:  0.6812171988617587
```

همینطور میتوان اطلاعات مدل را به طور کلی بر روی داده های تست بررسی کرد که به صورت زیر است :

همینطور که در این شکل نمایش داده میشود

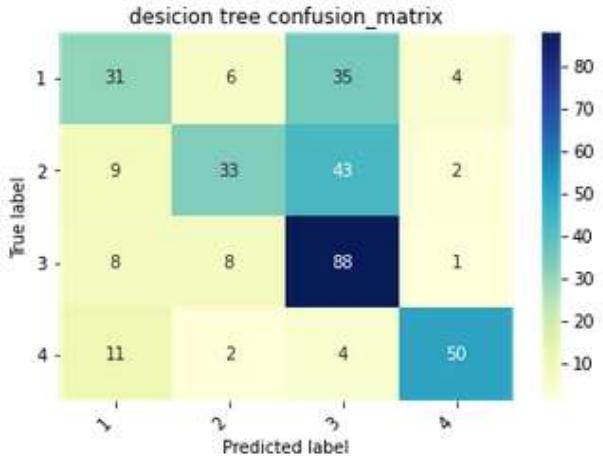
	precision	recall	f1-score	support
1	0.53	0.41	0.46	76
2	0.67	0.38	0.49	87
3	0.52	0.84	0.64	105
4	0.88	0.75	0.81	67
accuracy			0.60	335
macro avg	0.65	0.59	0.60	335
weighted avg	0.63	0.60	0.59	335

Precision نشان دهنده مقداری است که از بین کلیه مقادیری که به طور مثال به عنوان کلاس ۱ سنجیده شده اند چه تعداد درست بوده است. این مقدار برای کلاس ۴ که برخورد کردن به مانع و ایستادن خورد است بیشترین است (ینی کلاس چهارم به نسبت بقیه کلاس ها بهتر سنجیده شده است.)

مقدار recall نشان دهنده این است که از نمونه هایی که برچسب کلاس یک را داشند چه تعداد به درستی سنجیده شده اند که همانطور که در شکل میبینیم کلاس یک که به معنای بالا رفتن سرعت ناگهانی ماشین بود دقت پیشبینی که توسط این الگوریتم دارد نسبت به دو الگوریتم قبل بهتر شده و به ۴۱ رسیده است در حالی که در الگوریتم قبل ۰,۲۶ بود. و البته مقدار کلاس ۲ کاهش یافته است. همینطور برای کلاس ۳.

مقدار f1-score برابر 2*(prc*recall)/(pre+recall) نشان دهنده ی مقدار دقت کلی برای پیش بینی هر کلاس است که برای کلاس ۴ بیشترین دقت و برای کلاس ۱ کمترین دقت را داریم.

همینطور confusion matrix نتایج طبقه بندی شده به صورت شکل زیر است :



همانطور که در شکل مشاهده میشود مقداری که برای کلاس۳ بوده و به عنوان کلاس۳ تشخیص داده شده است نسبت به قسمت های قبلی افزایش یافته است و البته از طرفی هم مقداری که با عنوان کلاس ۳ شناخته شده و کلاس ۳ نبوده زیاد است و کلاس دو به نسبت الگوریتم های قبلی دقت پیشبینی آن کاهش یافته است.

و کلاس ۴ همانند نتیجه هایی که در الگوریتم های قبلی مشاهده کردیم به خوبی طبقه بندی شده است. و این امر بدیهی است زیرا این کلاس به معنی برخورد ناگهانی به یک سطح هست و طبیعی است که پارامتر های آن با دیگر پارامترها کمی متفاوت تر و قابل تشخیص‌تر باشد.

نتیجه گیری

در این گزارش متوجه آن شدیم که زمانی که مدل پیچیدگی زیادی دارد(مثلا مقدار عمق درخت در الگوریتم درخت تصمیم زیاد باشد) مدل بیشتر سعی میکند که داده ی train را حفظ کند و این اتفاق باعث overfit در طبقه بندی ما میشود.

همینطور نتیجه دیگری که از داده ها گرفتیم اینکه کلاس ۱ یا همان با سرعت حرکت کردن راننده به خوبی قابل تشخیص نیست و با کلاس های دیگر اشتباه طبقه بندی میشود . همینطور کلاس ۴ یا به مانع برخورد کردن راحت تر از بقیه کلاس ها طبقه بندی میشود به این دلیل که متغیر های شتاب برای آن به نسبت بقیه کلاسها متفاوت است.

و همینطور مشاهده کردیم که با تغیر دادن هایپر پارامتر های مختلف میتوان مدل بهتری بدست آورد.

منابع

(data.mendeley., 2020)

How to Identify Overfitting Machine Learning Models in Scikit-Learn, 2020)

sklearn.neighbors.KNeighborsClassifier¶, 2016)

how-to-tune-a-decision-tree, 2020)

improve-the-performance-of-a-machine-learning-model, 2019)