بسمه تعالی

نام درس:

کاوش دادگان انبوه (**Big data**)

نام پروژه:

----------------

نام استاد:

دکتر الهام آخوندزاده

نام **TA** درس:

مهندس نجفی

نام و شماره دانشجویی اعضای گروه:

سمیه حمیدی 40066

هاله خوش شانس 40066

جواد رفیعی فرد 40066241005

محمدرضا شاقوزی 40066

بسمه تعالی

گزارش پروژه درس داده کاوی

ویرایش: 20/11/1401 – تدوین: 06/11/1401 – تهیه‏کننده: سمیه حمیدی، هاله خوش شانس، جواد رفیعی فرد، محمدرضا شاقوزی

**فهرست**

[1 معرفی مجموعه داده و توصیف آن (رفیعی فرد) 5](#_Toc108903338)

[1-1 مجموعه داده در یک نگاه (رفیعی فرد) 5](#_Toc108903339)

[1-2 متغیرهای ورودی (رفیعی فرد) 6](#_Toc108903340)

[1-3 متغیر خروجی (رفیعی فرد) 20](#_Toc108903341)

[2 مرحله پیش پردازش (سلامی) 21](#_Toc108903342)

[2-1 تبدیل به داده‌های عددی (سلامی) 21](#_Toc108903343)

[2-2 نرمال‌‌سازی (سلامی) 22](#_Toc108903344)

[2-2-1 نرمال‌سازی با استفاده از روش MIN\_MAX (سلامی) 22](#_Toc108903345)

[2-2-2 نرمال‌سازی با استفاده از روش ZSCORE (رفیعی فرد) 23](#_Toc108903346)

[2-3 تشخیص و حذف داده های پرت (سلامی) 23](#_Toc108903347)

[2-3-1 تشخیص داده های پرت (سلامی) 23](#_Toc108903348)

[2-3-2 حذف داده های پرت با استفاده از IQR (رفیعی-سلامی) 25](#_Toc108903349)

[2-3-3 حذف اوت‌لایرها با استفاده از ZSCORE (سلامی) 26](#_Toc108903350)

[2-4 بررسی مقادیر از دست رفته (سلامی) 27](#_Toc108903351)

[2-4-1 ایجاد مقادیر از دست رفته مصنوعی (سلامی) 28](#_Toc108903352)

[2-4-2 پر کردن مقادیر از دست رفته و بررسی آن‌ها (رفیعی فرد-سلامی) 28](#_Toc108903353)

[2-5 انتخاب متغیر‌ها (سلامی) 29](#_Toc108903354)

[2-5-1 انتخاب متغیر با random forest (سلامی) 29](#_Toc108903355)

[2-5-2 انتخاب متغیر با Boruta (رفیعی فرد) 30](#_Toc108903356)

[3 ساخت مدل (رفیعی فرد) 31](#_Toc108903357)

[3-1 آماده سازی داده آموزشی و تست (رفیعی فرد) 31](#_Toc108903358)

[3-2 معیار اصلی جهت انتخاب بهترین مدل (رفیعی فرد) 31](#_Toc108903359)

[3-3 ساخت مدل با درخت تصمیم و تنظیم هایپرپارامترهای آن (رفیعی فرد) 34](#_Toc108903360)

[3-3-1 معرفی هایپرپارمترها (رفیعی فرد) 34](#_Toc108903361)

[3-3-2 تنظیم پارامترها برای به دست آوردن بهترین حالت بر حسب معیار Recall (رفیعی فرد) 34](#_Toc108903362)

[3-3-3 ارزیابی مدل بر روی داده آموزشی و تست (رفیعی فرد) 35](#_Toc108903363)

[3-3-4 جستجوی کمترین حالت overfitting (رفیعی فرد) 36](#_Toc108903364)

[3-3-5 بررسی تغییر نتایج با تغییر هایپرپارامترها (رفیعی فرد) 37](#_Toc108903365)

[3-3-6 بهترین هایپرپارامترها و رسم نمودار نتایج بهینه (رفیعی فرد) 39](#_Toc108903366)

[3-4 ساخت مدل با **Bagging** و مدل پایه درخت تصمیم و تنظیم هایپرپارامترهای آن (رفیعی فرد) 42](#_Toc108903367)

[3-4-1 معرفی هایپرپارمترها (رفیعی فرد) 42](#_Toc108903368)

[3-4-2 تنظیم پارامترها برای به دست آوردن بهترین حالت بر حسب معیار Recall (رفیعی فرد) 42](#_Toc108903369)

[3-4-3 ارزیابی مدل بر روی داده آموزشی و تست (رفیعی فرد) 43](#_Toc108903370)

[3-4-4 جستجوی کمترین حالت overfitting (رفیعی فرد) 44](#_Toc108903371)

[3-4-5 بررسی تغییر نتایج با تغییر هایپرپارامترها (رفیعی فرد) 46](#_Toc108903372)

[3-4-6 بهترین هایپرپارامترها و رسم نمودار نتایج بهینه (رفیعی فرد) 52](#_Toc108903373)

[3-5 **ساخت مدل با Bagging و مدل پایه KNN و تنظیم هایپرپارامترها** (رفیعی فرد) 54](#_Toc108903374)

[3-5-1 معرفی هایپرپارمترها (رفیعی فرد) 54](#_Toc108903375)

[3-5-2 تنظیم پارامترها برای به دست آوردن بهترین حالت بر حسب معیار Recall (رفیعی فرد) 55](#_Toc108903376)

[3-5-3 ارزیابی مدل بر روی داده آموزشی و تست (رفیعی فرد) 56](#_Toc108903377)

[3-5-4 جستجوی کمترین حالت overfitting (رفیعی فرد) 57](#_Toc108903378)

[3-5-5 بررسی تغییر نتایج با تغییر هایپرپارامترها (رفیعی فرد) 58](#_Toc108903379)

[3-5-6 بهترین هایپرپارامترها و رسم نمودار نتایج بهینه (رفیعی فرد) 63](#_Toc108903380)

[3-6 **ساخت مدل با جنگل تصادفی و تنظیم هایپرپارامترها** (رفیعی فرد) 66](#_Toc108903381)

[3-6-1 معرفی هایپرپارمترها (رفیعی فرد) 66](#_Toc108903382)

[3-6-2 تنظیم پارامترها برای به دست آوردن بهترین حالت بر حسب معیار Recall (رفیعی فرد) 66](#_Toc108903383)

[3-6-3 ارزیابی مدل بر روی داده آموزشی و تست (رفیعی فرد) 67](#_Toc108903384)

[3-6-4 جستجوی کمترین حالت overfitting (رفیعی فرد) 69](#_Toc108903385)

[3-6-5 بررسی تغییر نتایج با تغییر هایپرپارامترها (رفیعی فرد) 70](#_Toc108903386)

[3-6-6 بهترین هایپرپارامترها و رسم نمودار نتایج بهینه (رفیعی فرد) 72](#_Toc108903387)

[3-7 **ساخت مدل با آدابوست و تنظیم هایپرپارامترها** (رفیعی فرد) 75](#_Toc108903388)

[3-7-1 معرفی هایپرپارمترها (رفیعی فرد) 75](#_Toc108903389)

[3-7-2 تنظیم پارامترها برای به دست آوردن بهترین حالت بر حسب معیار Recall (رفیعی فرد) 75](#_Toc108903390)

[3-7-3 ارزیابی مدل بر روی داده آموزشی و تست (رفیعی فرد) 76](#_Toc108903391)

[3-7-4 جستجوی کمترین حالت overfitting (رفیعی فرد) 77](#_Toc108903392)

[3-7-5 بررسی تغییر نتایج با تغییر هایپرپارامترها (رفیعی فرد) 79](#_Toc108903393)

[3-7-6 بهترین هایپرپارامترها و رسم نمودار نتایج بهینه (رفیعی فرد) 83](#_Toc108903394)

[3-8 **ساخت مدل با KNN و تنظیم هایپرپارامترها** (رفیعی فرد) 86](#_Toc108903395)

[3-8-1 معرفی هایپرپارمترها (رفیعی فرد) 86](#_Toc108903396)

[3-8-2 تنظیم پارامترها برای به دست آوردن بهترین حالت بر حسب معیار Recall (رفیعی فرد) 86](#_Toc108903397)

[3-8-3 ارزیابی مدل بر روی داده آموزشی و تست (رفیعی فرد) 87](#_Toc108903398)

[3-8-4 بررسی تغییر نتایج با تغییر هایپرپارامترها (رفیعی فرد) 88](#_Toc108903399)

[3-8-5 بهترین هایپرپارامترها و رسم نمودار نتایج بهینه (رفیعی فرد) 91](#_Toc108903400)

[3-9 **ساخت مدل با SVM و تنظیم هایپرپارامترها** (رفیعی فرد) 93](#_Toc108903401)

[3-9-1 معرفی هایپرپارمترها (رفیعی فرد) 93](#_Toc108903402)

[3-9-2 تنظیم پارامترها برای به دست آوردن بهترین حالت بر حسب معیار Recall (رفیعی فرد) 93](#_Toc108903403)

[3-9-3 ارزیابی مدل بر روی داده آموزشی و تست (رفیعی فرد) 94](#_Toc108903404)

[3-9-4 جستجوی کمترین حالت overfitting (رفیعی فرد) 95](#_Toc108903405)

[3-9-5 بررسی تغییر نتایج با تغییر هایپرپارامترها (رفیعی فرد) 97](#_Toc108903406)

[3-9-6 بهترین هایپرپارامترها و رسم نمودار نتایج بهینه (رفیعی فرد) 98](#_Toc108903407)

[4 مقایسه دسته بندها(رفیعی فرد-سلامی) 101](#_Toc108903408)

[5 تفسیر نتایج(رفیعی فرد) 101](#_Toc108903409)

# معرفی مجموعه داده و توصیف آن (رفیعی فرد)

در این تحقیق قصد داریم یک پروژه داده کاوی را بر روی مجموعه داده بیمه خودرو انجام دهیم. این مجموعه داده که از سایت kaggle استخراج شده مربوط به یک شرکت بیمه می باشد و در آدرس زیر قرار دارد:

<https://www.kaggle.com/datasets/sagnik1511/car-insurance-data>

و دارای 19 ویژگی (ستون) و 10،000 رکورد (سطر) است. که هر رکورد حاوی اطلاعات مربوط به یک شخص شامل سن، جنسیت، تحصیلات و ... است. 18 ویژگی به عنوان متغیرهای ورودی و یک ویژگی متغیر خروجی است که مشخص می کند آیا شخص خسارتی را از بیمه مطالبه کرده است یا خیر به عبارت دیگر آیا شخص از بیمه اتومبیل خود استفاده کرده است یا خیر. اطلاعات کلی این مجموعه داده به صورت زیر است:

## مجموعه داده در یک نگاه (رفیعی فرد)

عناوین ویژگی ها (ستون ها):

['ID', 'AGE', 'GENDER', 'RACE', 'DRIVING\_EXPERIENCE', 'EDUCATION',

'INCOME', 'CREDIT\_SCORE', 'VEHICLE\_OWNERSHIP', 'VEHICLE\_YEAR',

'MARRIED', 'CHILDREN', 'POSTAL\_CODE', 'ANNUAL\_MILEAGE', 'VEHICLE\_TYPE',

'SPEEDING\_VIOLATIONS', 'DUIS', 'PAST\_ACCIDENTS', 'OUTCOME']

تعداد مقادیر هر ستون، نوع داده آنها:

RangeIndex: 10000 entries, 0 to 9999

Data columns (total 19 columns):

# Column Non-Null Count Dtype

--- ------ -------------- -----

0 ID 10000 non-null int64

1 AGE 10000 non-null object

2 GENDER 10000 non-null object

3 RACE 10000 non-null object

4 DRIVING\_EXPERIENCE 10000 non-null object

5 EDUCATION 10000 non-null object

6 INCOME 10000 non-null object

7 CREDIT\_SCORE 9018 non-null float64

8 VEHICLE\_OWNERSHIP 10000 non-null int64

9 VEHICLE\_YEAR 10000 non-null object

10 MARRIED 10000 non-null int64

11 CHILDREN 10000 non-null int64

12 POSTAL\_CODE 10000 non-null int64

13 ANNUAL\_MILEAGE 9043 non-null float64

14 VEHICLE\_TYPE 10000 non-null object

15 SPEEDING\_VIOLATIONS 10000 non-null int64

16 DUIS 10000 non-null int64

17 PAST\_ACCIDENTS 10000 non-null int64

18 OUTCOME 10000 non-null int64

dtypes: float64(2), int64(9), object(8)

memory usage: 1.4+ MB

ملاحظه می شود که ستون شماره 7 (CREDIT\_SCORE) و ستون شماره 13 (ANNUAL\_MILEAGE) دارای تعدادی سلول خالی (Nan) می باشند ولی بقیه ستون ها کامل هستند.

تعداد مقادیر یکتای ستون ها:

ID 10000

AGE 4

GENDER 2

RACE 2

DRIVING\_EXPERIENCE 4

EDUCATION 3

INCOME 4

CREDIT\_SCORE 9018

VEHICLE\_OWNERSHIP 2

VEHICLE\_YEAR 2

MARRIED 2

CHILDREN 2

POSTAL\_CODE 4

ANNUAL\_MILEAGE 21

VEHICLE\_TYPE 2

SPEEDING\_VIOLATIONS 21

DUIS 7

PAST\_ACCIDENTS 15

OUTCOME 2

در ادامه به معرفی ویژگی ها و توصیف داده بر اساس آن ها می پردازیم.

## متغیرهای ورودی (رفیعی فرد)

1. شناسه (ID): از نوع عددی و برای هر شخص منحصر به فرد است
2. سن (AGE): از نوع ترتیبی است و در یکی از بازه های زیر است:

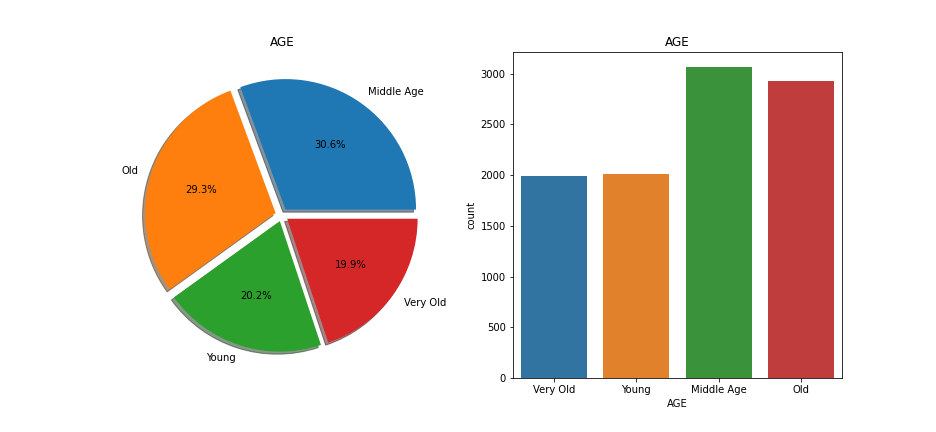
['16-25', '26-39', '40-64', '65+']

که مقادیر آن را به مقادیر زیر تغییر نام دادیم:

['Young', 'Middle Age', 'Old', 'Very Old'}

که به ترتیب به معنی جوان، میانسال، پیر و خیلی پیر است.

نمودار توزیع این ویژگی به صورت زیر است:

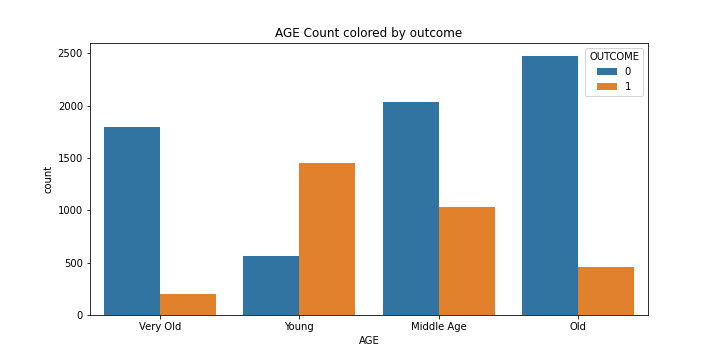
 Middle Age 3063

Old 2931

Young 2016

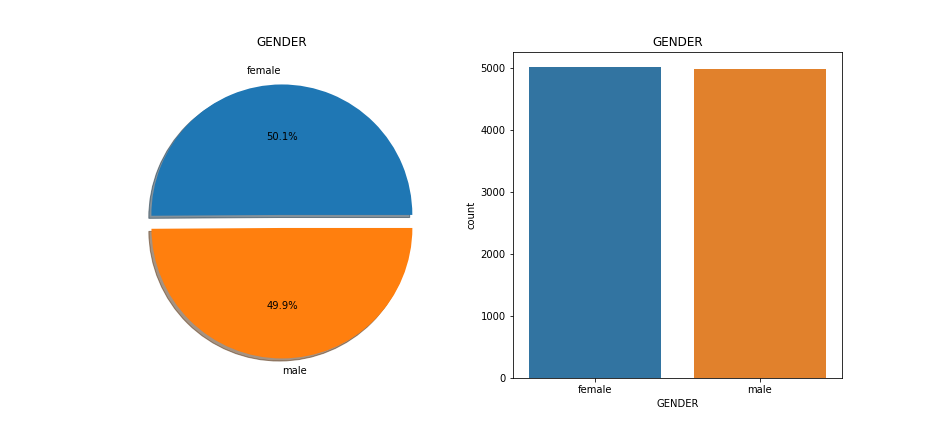
Very Old 1990

توزیع آن بر حسب دو کلاس به صورت زیر می باشد:



1. جنسیت (GENDER): از نوع باینری است که دارای مقادیر ['male', 'female'] به معنی آقا و خانم است.

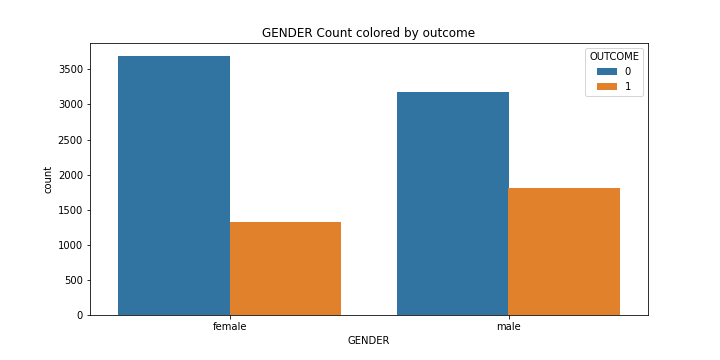
نمودار توزیع این ویژگی به صورت زیر است:



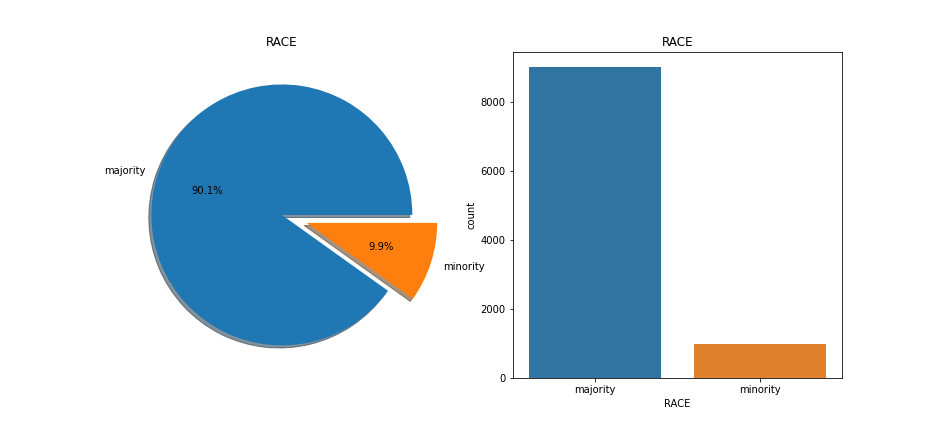
female 5010

male 4990

توزیع این ویژگی بر حسب دو کلاس به صورت زیر می باشد:

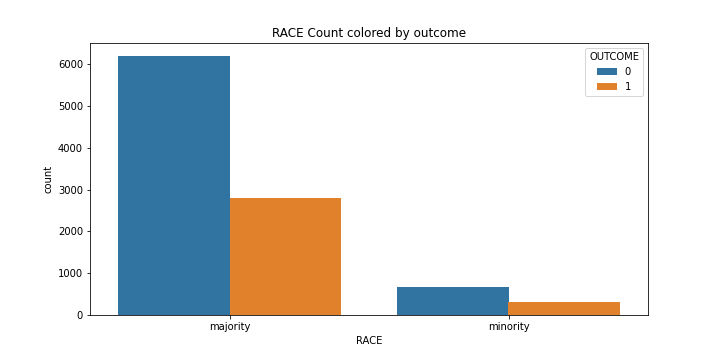


1. نژاد (RACE): از نوع باینری است که دارای مقادیر ['majority' 'minority'] به معنی اکثریت و اقلیت تقسیم می گردد و نمودار توزیع آن به صورت زیر است:

 majority 9012

minority 988

توزیع این ویژگی بر حسب دو کلاس به صورت زیر می باشد:



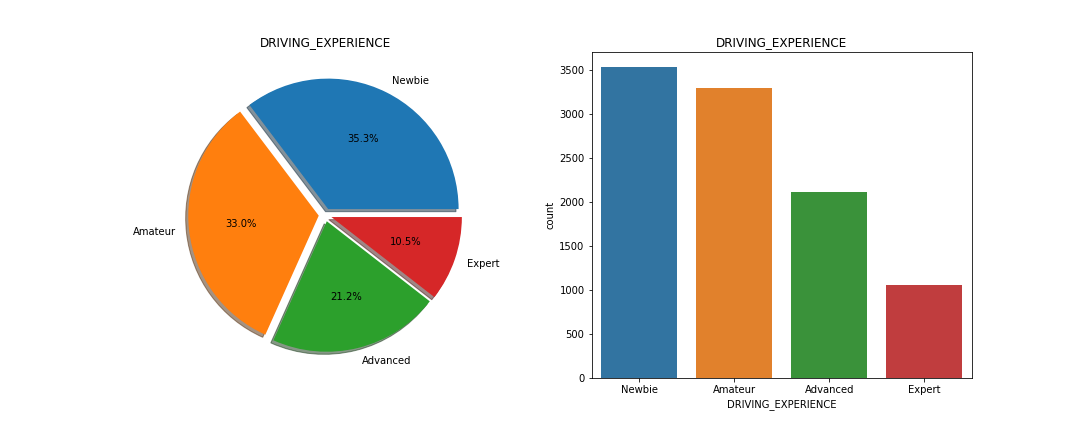
1. مهارت رانندگی (DRIVING\_EXPERIENCE): از نوع ترتیبی و در یکی از بازه های زیر است:

['0-9y', '10-19y', '20-29y', '30y+']

که ما آن ها را به صورت زیر نام گذاری کردیم:

{'0-9y': 'Newbie', '10-19y': 'Amateur', '20-29y': 'Advanced', '30y+': 'Expert'}

که به ترتیب به معنی تازه کار، آماتور، پیشرفته و حرفه ای است. نمودار توزیع آن به صورت زیر است:

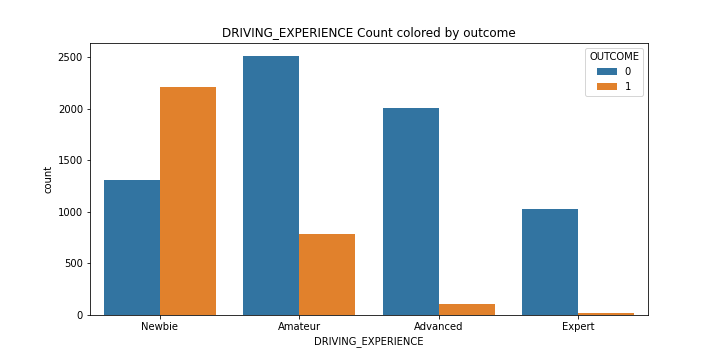
 Newbie 3530

Amateur 3299

Advanced 2119

Expert 1052

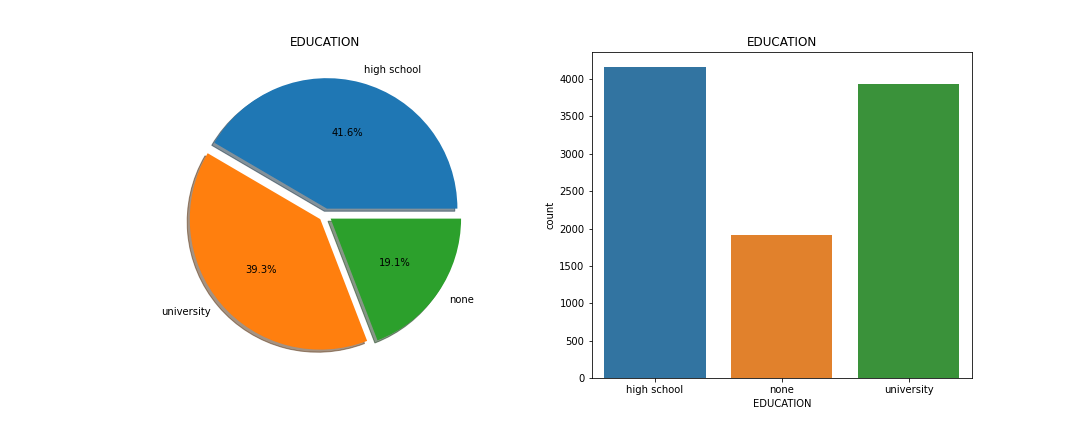
توزیع این ویژگی بر حسب دو کلاس به صورت زیر می باشد:



1. تحصیلات (EDUCATION): از نوع ترتیبی و دارای یکی از مقادیر زیر است:

['none', 'high school', 'university']

نمودار توزیع آن به صورت زیر است:

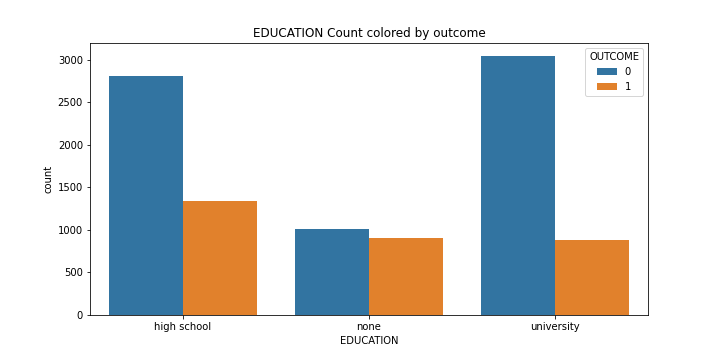


high school 4157

university 3928

none 1915

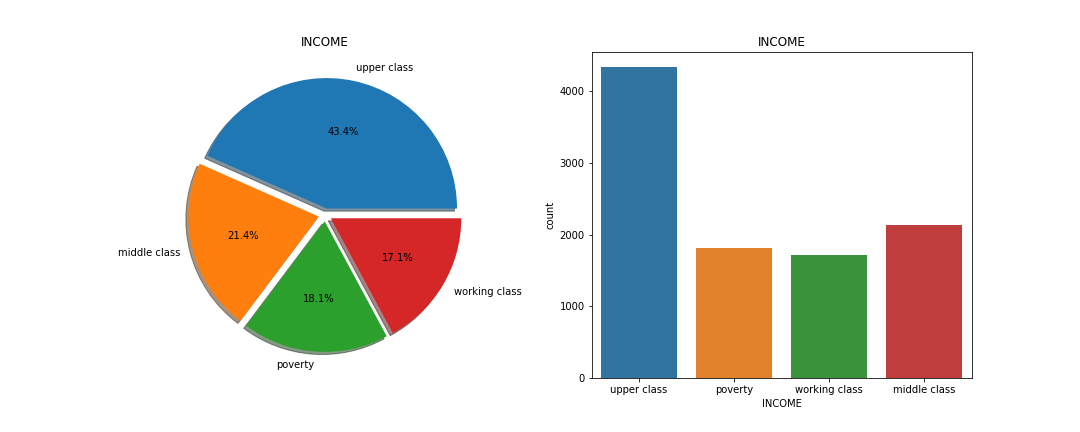
توزیع این ویژگی بر حسب دو کلاس به صورت زیر می باشد:



1. درآمد (INCOME): از نوع ترتیبی و دارای یکی از مقادیر زیر است:

['poverty', 'working class', 'middle class', 'upper class']

که به ترتیب به معنی فقیر، کارگر، قشر متوسط و قشر مرفه است و نمودار توزیع آن به صورت زیر است:

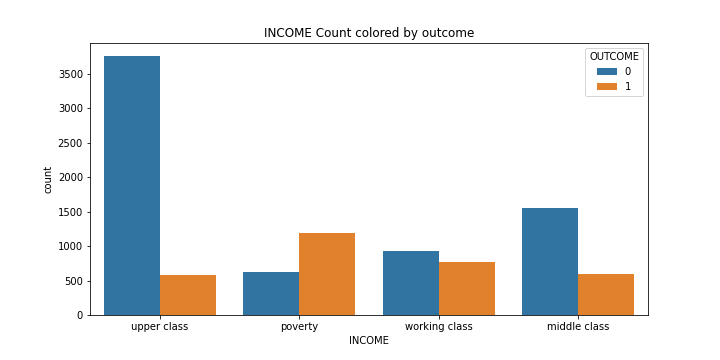
 upper class 4336

middle class 2138

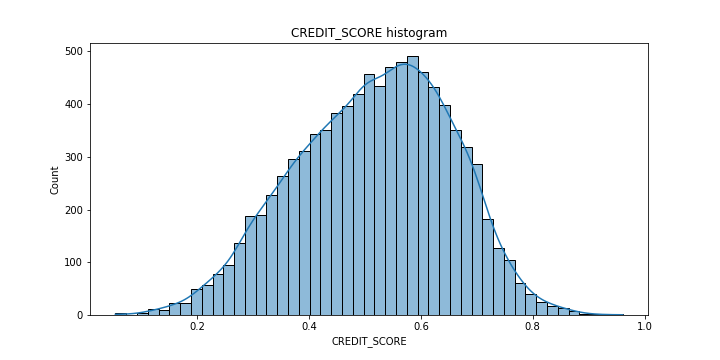
poverty 1814

working class 1712

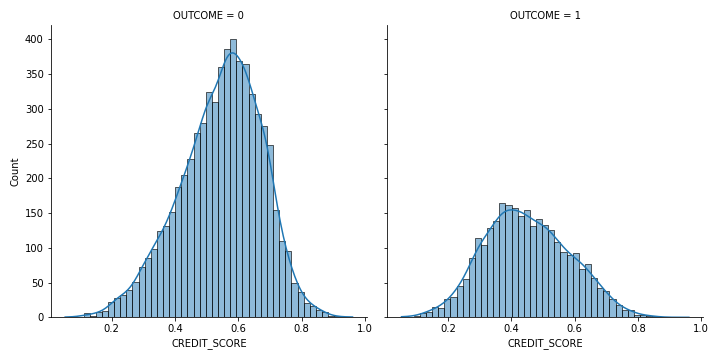
توزیع این ویژگی بر حسب دو کلاس به صورت زیر می باشد:



1. اعتبار (CREDIT\_SCORE): از نوع عددی اعشاری (float) و پیوسته است و مقداری بین 0 و 1 دارد و نمودار توزیع آن به صورت زیر است:



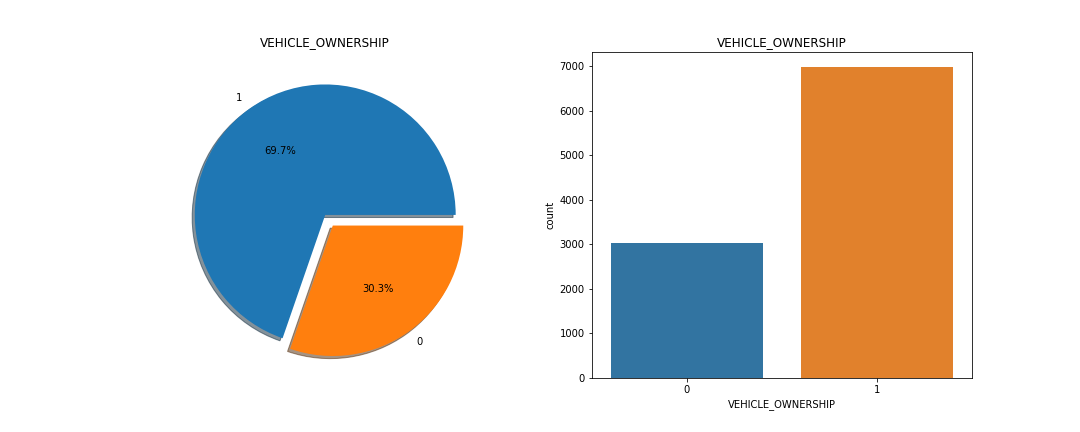
توزیع این ویژگی بر حسب دو کلاس به صورت زیر می باشد:



1. مالکیت خودرو (VEHICLE\_OWNERSHIP): از نوع باینری است و دارای یکی از مقادیر 0 به معنی اینکه مالک خودرو نیست یا 1 به معنی اینکه مالک خودرو است می باشد.

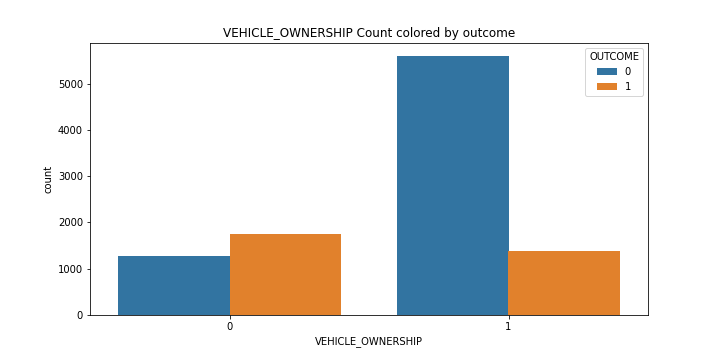
توضیح: با توجه به اینکه مجموعه داده در مورد بیمه خودرو است بنابراین مقدار 0 در این ویژگی به معنی این است که خودرو در تصرف فرد است ولی او مالک خودرو نیست مانند راننده تاکسی یا کسانی که خودرو را اجاره کرده اند یا به هر نحوی از خودرو شخص دیگری استفاده می کنند.

نمودار توزیع آن به صورت زیر است:

 1 6970

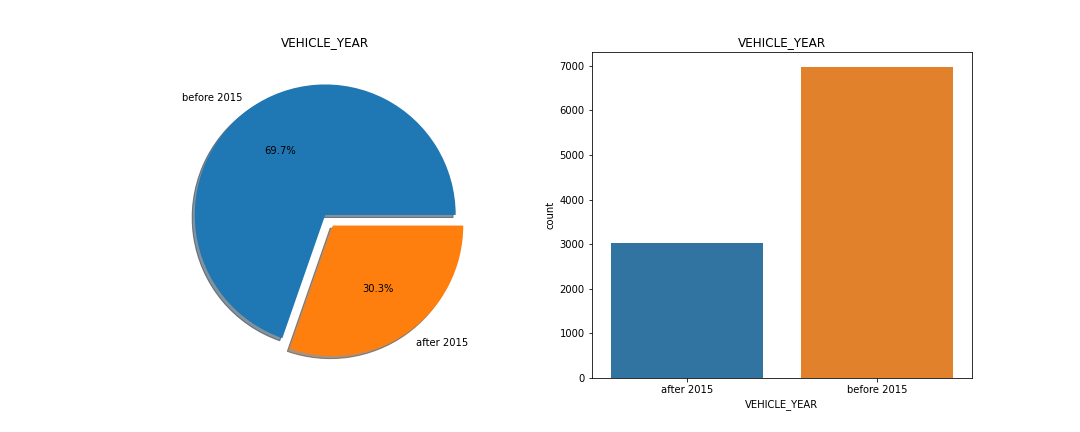
0 3030

توزیع این ویژگی بر حسب دو کلاس به صورت زیر می باشد:



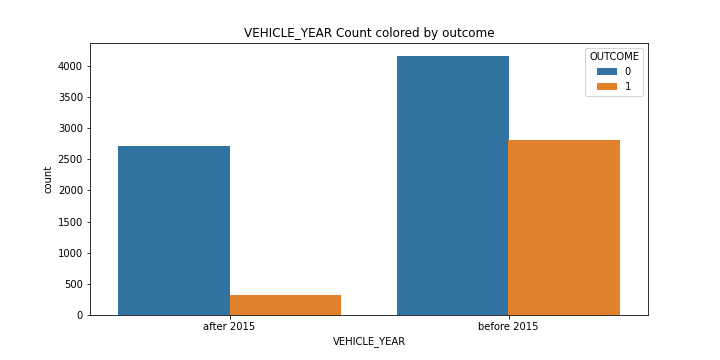
1. سال تولید خودرو (VEHICLE\_YEAR): از نوع باینری و دارای یکی از مقادیر زیر است:

[befor 2015', 'after 2015''] که به معنی این است که قبل از سال 2015 تولید شده یا بعد از آن و نمودار توزیع آن به صورت زیر است:

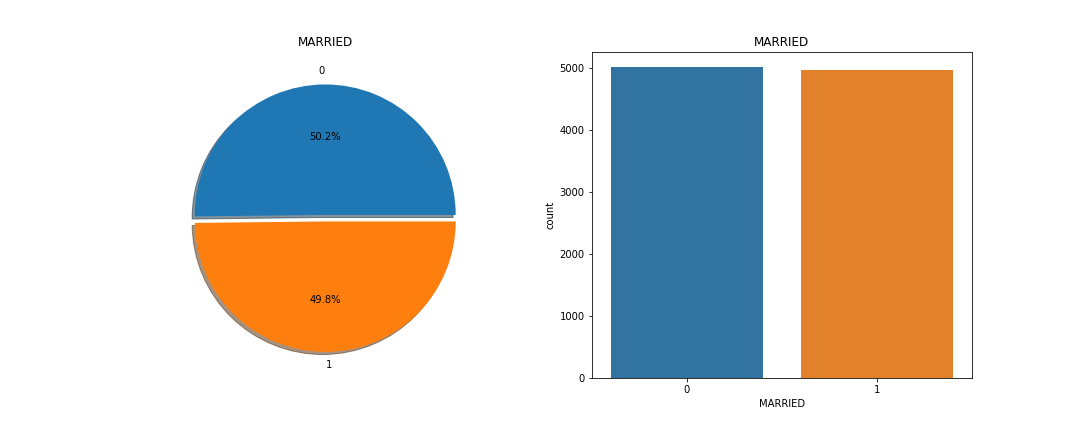
 before 2015 6967

after 2015 3033

توزیع این ویژگی بر حسب دو کلاس به صورت زیر می باشد:

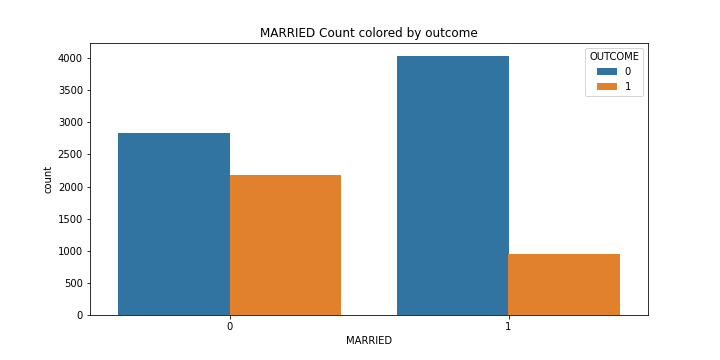


1. وضعیت تأهل (MARRIED): از نوع باینری است و دارای یکی از مقادیر 0 به معنی مجرد و 1 به معنی متأهل است و نمودار توزیع آن به صورت زیر است:

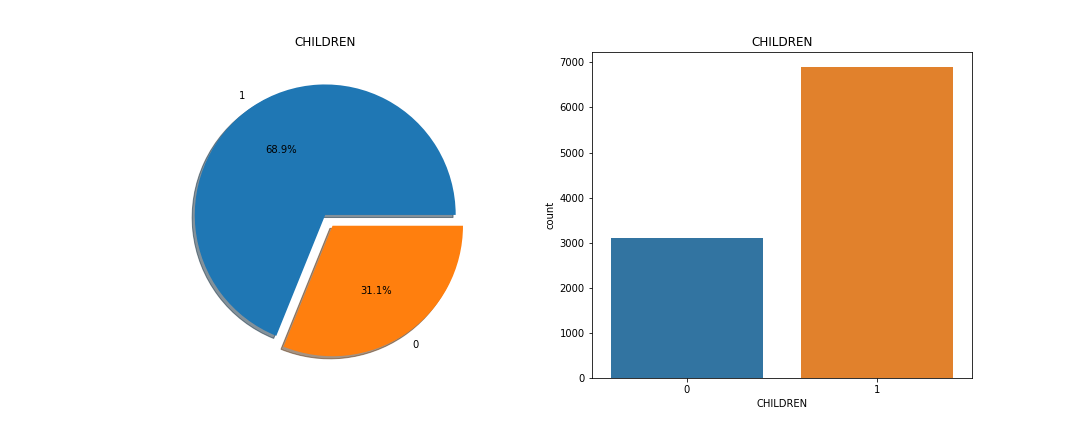
 0 5018

1 4982

توزیع این ویژگی بر حسب دو کلاس به صورت زیر می باشد:



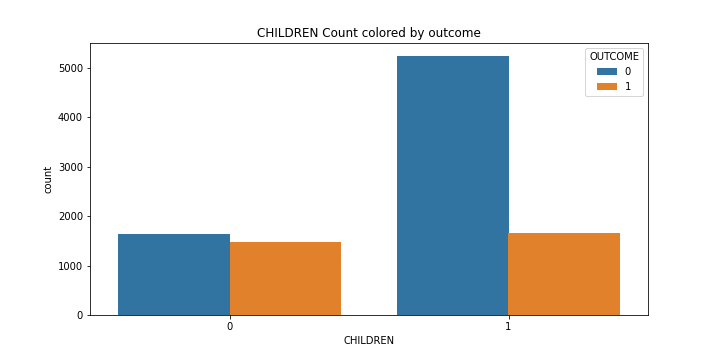
1. وضعیت فرزند (CHILDREN): از نوع باینری است و دارای یکی از مقادیر 0 به معنی عدم داشتن فرزند و 1 به معنی داشتن فرزند است و نمودار توزیع آن به صورت زیر است:



1 6888

0 3112

توزیع این ویژگی بر حسب دو کلاس به صورت زیر می باشد:



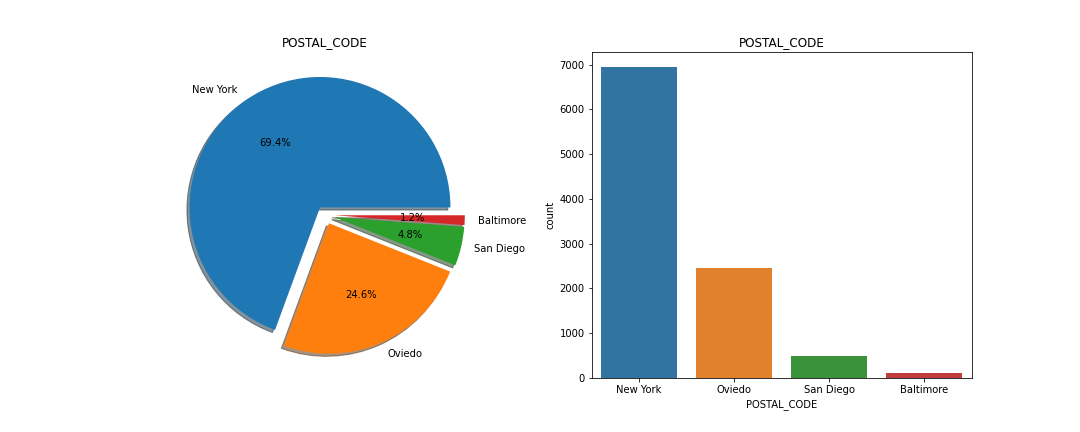
1. محل سکونت (POSTAL\_CODE): از نوع اسمی است و دارای یکی از مقادیر زیر است:

['10238', 21217', '32765', '92101']

توضیح: عنوان این ویژگی کدپستی ذکر شده اما با توجه به اینکه 5 رقمی است احتمالاً 5 رقم اول کدپستی است که بیشتر بیانگر منطقه محل سکونت است تا کدپستی. بهتر بود بجای عنوان کدپستی از عنوان دیگری مثل REGION استفاده می کرد. با جستجو در اینترنت متوجه شدیم که این کدپستی ها به ترتیب متعلق به مناطق زیر هستند:

['New York', ' Baltimore', 'Oviedo', 'San Diego']

نمودار توزیع این ویژگی به صورت زیر است:

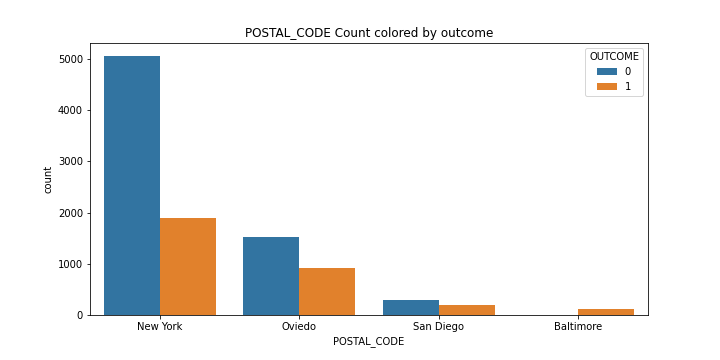
 New York 6940

Oviedo 2456

San Diego 484

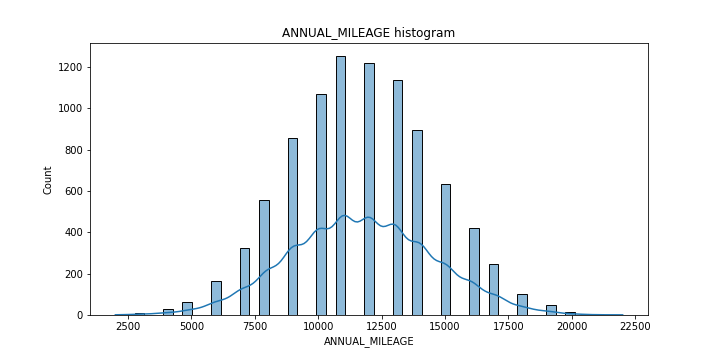
Baltimore 120

و توزیع این ویژگی بر حسب دو کلاس به صورت زیر می باشد:

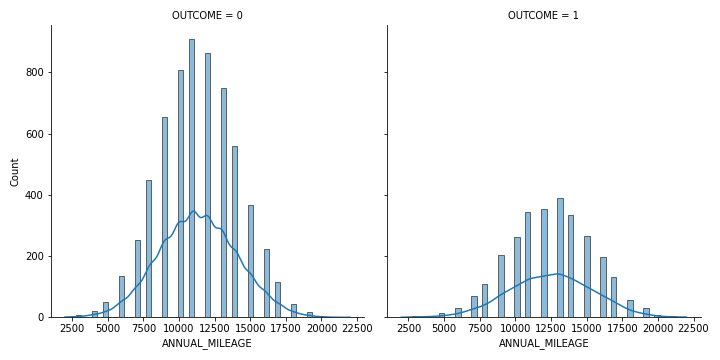


1. مسافت پیموده شده سالیانه (ANNUAL\_MILEAGE): از نوع عددی صحیح (int) و مقدار آن بین 2000 تا 22000 است.

نمودار توزیع آن به صورت زیر است:



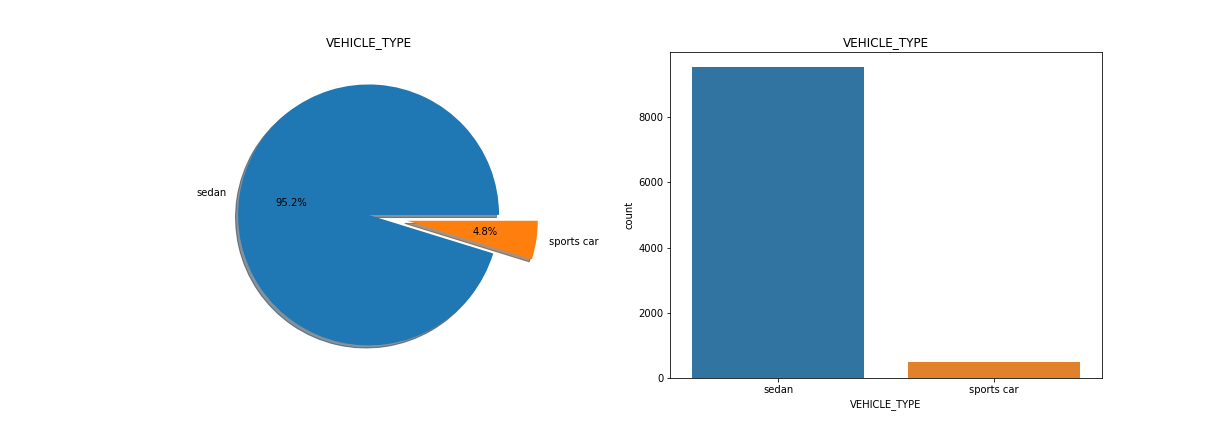
و توزیع این ویژگی بر حسب دو کلاس به صورت زیر می باشد:



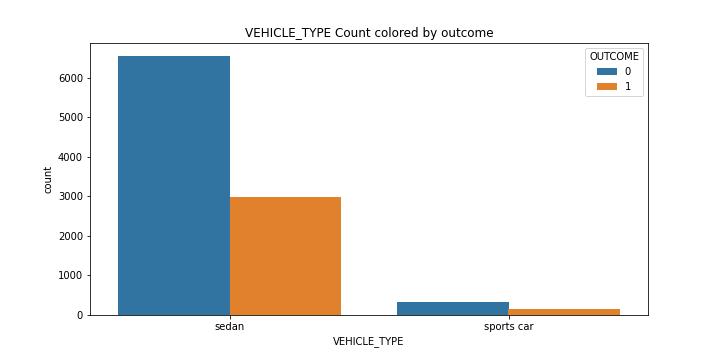
1. نوع خودرو (VEHICLE\_TYPE): از نوع باینری و دارای یکی از مقادیر زیر است:

['sedan', 'sports car'] که مربوط به خودروهای معمولی و خودروهای مسابقه ای می باشد.

نمودار توزیع آن به صورت زیر است:

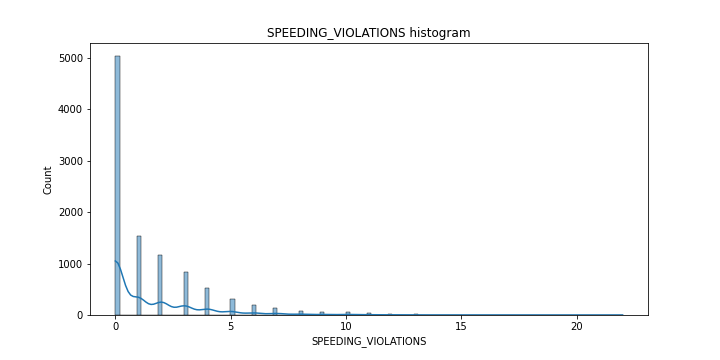


و توزیع این ویژگی بر حسب دو کلاس به صورت زیر می باشد:

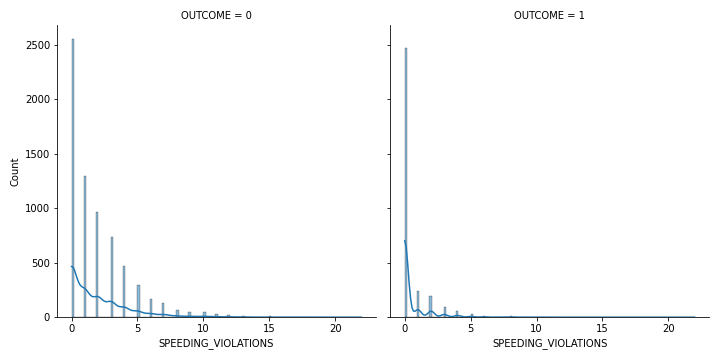


1. تخلف سرعت (SPEEDING\_VIOLATIONS): از نوع عددی صحیح (int) است که مقدار آن بین 0 و 22 تغییر می کند و بیانگر تعداد دفعات تعدی راننده از سرعت مجاز است.

نمودار توزیع آن به صورت زیر است:

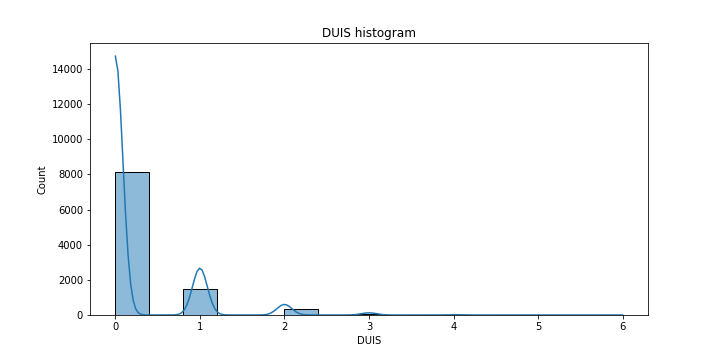


و توزیع این ویژگی بر حسب دو کلاس به صورت زیر می باشد:

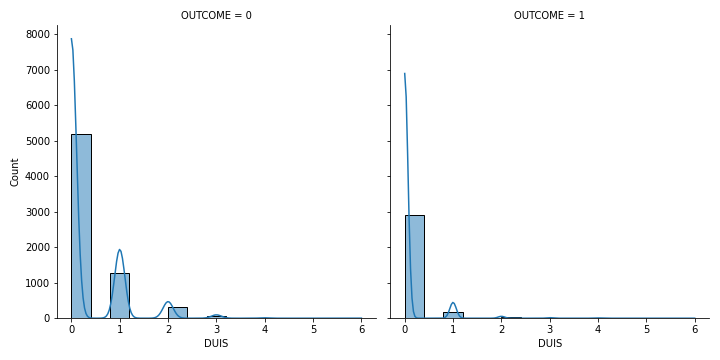


1. تخلف مصرف مشروبات الکلی (DUIS): از نوع عددی صحیح (int) و مقدار آن بین 0 تا 6 متغیر است.

نمودار توزیع آن به صورت زیر است:

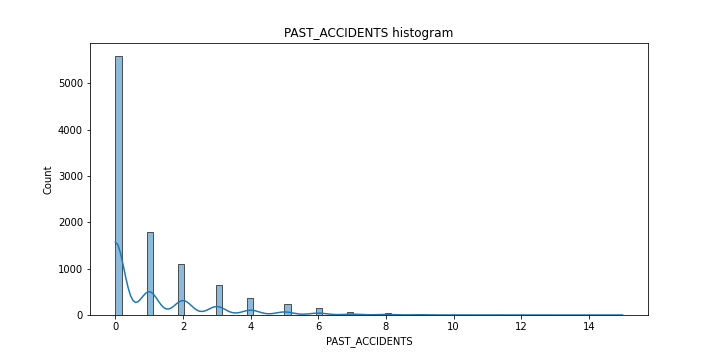


و توزیع این ویژگی بر حسب دو کلاس به صورت زیر می باشد:

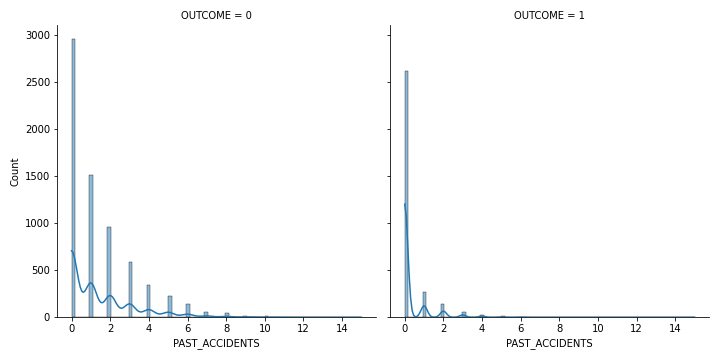


1. تعداد تصادفات گذشته (PAST\_ACCIDENTS): از نوع عددی صحیح (int) و مقدار آن بین 0 تا 15 است.

نمودار توزیع آن به صورت زیر است:

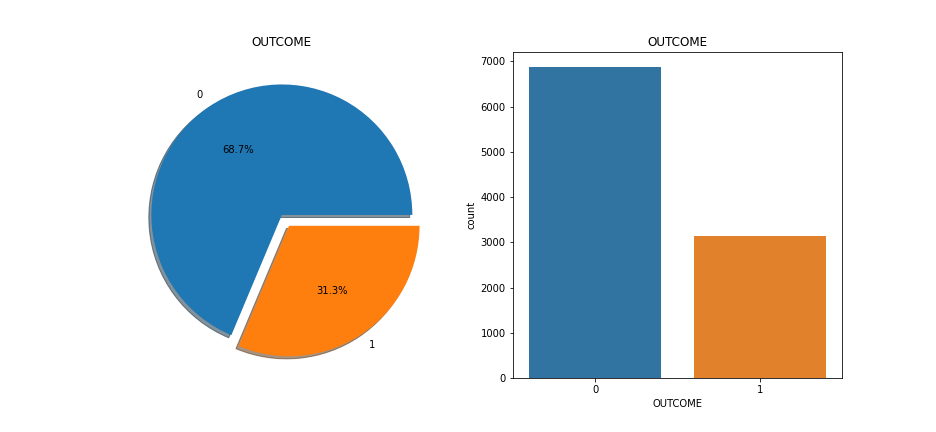


و توزیع این ویژگی بر حسب دو کلاس به صورت زیر می باشد:



## متغیر خروجی (رفیعی فرد)

نتیجه (OUTCOME): از نوع باینری است و دارای دو کلاس 0 و 1 است. این ویژگی، متغیر خروجی یا پاسخ این مجموعه داده است و بیانگر این موضوع است که آیا فرد از بیمه خود استفاده کرده است یا خیر. به عبارت دیگر آیا فرد از بیمه مطالبه خسارت کرده است یا نه. عدد 0 نشان دهنده عدم مطالبه خسارت و 1 نشان دهنده مطالبه خسارت می باشد. توزیع کلاس به صورت زیر می باشد:



از نمودار چنین برداشت می شود که این داده نامتقارن است که برای ساخت مدل لازم است با روش های مناسب آن را متقارن کنیم که در جای خود به آن خواهیم پرداخت.

# فراخوانی کتابخانه های مورد نیاز و انجام تنظیمات **pyspark**

ابتدا توسط کد زیر متدها و کتابخانه های مورد نیاز را فراخوانی نمودیم. همچنین تنظیمات اولیه مورد نیاز برای کتابخانه pyspark را انجام دادیم:

from pyspark.sql import SparkSession

from pyspark import SparkConf, SparkContext

from pyspark.sql.types import StructType,IntegerType,FloatType,BooleanType,StringType

from pyspark.sql.functions import rand, count, isnull, when, col

conf = SparkConf().setMaster("local[\*]").setAppName("My App")

sc = SparkContext.getOrCreate(conf = conf)

sc.\_conf.set('spark.executor.memory','15g')\

    .set('spark.driver.memory','15g')\

        .set('spark.driver.maxResultsSize','0')

spark=SparkSession.builder\

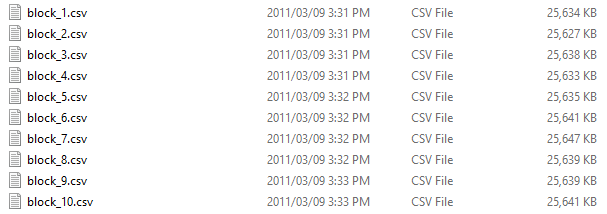
    .appName('myApp')\

        .config("spark.driver.memory", "15g")\

            .getOrCreate()

# تجمیع چانک های مجموعه داده

مجموعه داده شامل ده فایل با مشخصات زیر می باشد:



این چانک ها را توسط کد زیر تجمیع کرده و در یک متغیر دیتافریم ذخیره کردیم:

def load\_data(files,schema):

    df=spark.read.csv(files,header=True

                  ,schema=schema)

    return df

def load\_record\_linkage\_data():

    schema = StructType() \

      .add("id\_1",IntegerType(),True) \

      .add("id\_2",IntegerType(),True) \

      .add("cmp\_fname\_c1",FloatType(),True) \

      .add("cmp\_fname\_c2",FloatType(),True) \

      .add("cmp\_lname\_c1",FloatType(),True) \

      .add("cmp\_lname\_c2",FloatType(),True) \

      .add("cmp\_sex",IntegerType(),True) \

      .add("cmp\_bd",IntegerType(),True) \

      .add("cmp\_bm",IntegerType(),True) \

      .add("cmp\_by",IntegerType(),True) \

      .add("cmp\_plz",IntegerType(),True) \

      .add("is\_match",BooleanType(),False)

    files=[f'./data/block\_{id}.csv' for id in range(1,11)]

    return load\_data(files,schema=schema)

df=load\_record\_linkage\_data()

# تحلیل اکتشافی داده (**Exploratory data analysis**)

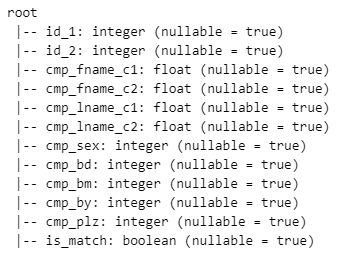
قرار هست خانم حمیدی زحمت این قسمت را بکشند 😊

# مرحله پیش پردازش

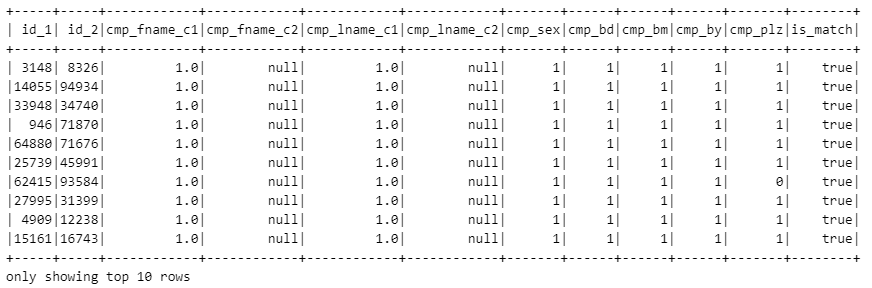
## تبدیل به داده‌های عددی

تمام ویژگی های این مجموعه داده از قبل عددی شده اند بجز خروجی که به صورت منطقی و باینری است:

df.printSchema()



df.show(10)



و آن را توسط تابع زیر عددی کردیم:

def convert\_label\_binary(input\_df):

    temp = input\_df.withColumn('label',

                             when(input\_df['is\_match']==True,

                                  lit(1)).otherwise(0)

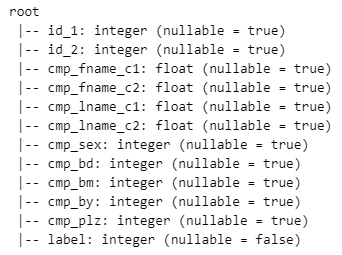
                                  )

    return temp

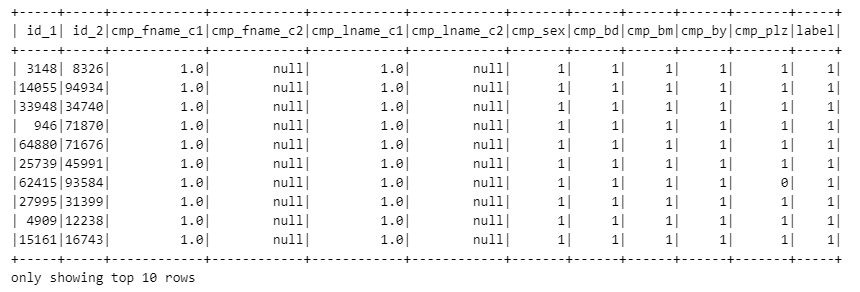
numerical\_df = convert\_label\_binary(df).drop('is\_match')

که نتیجه آن به صورت زیر می باشد:

numerical\_df.printSchema()



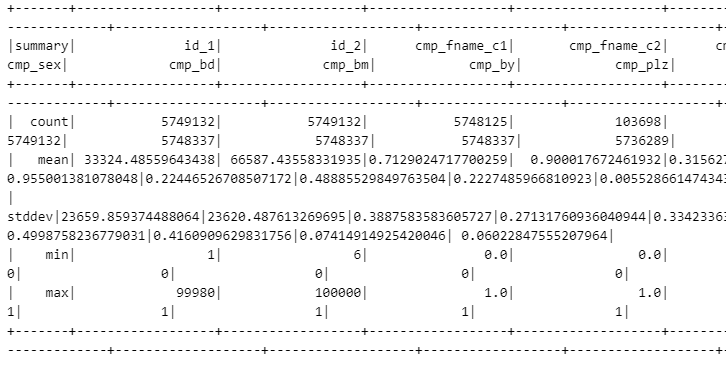
numerical\_df.show(10)



## نرمال سازی

ویژگی ها همه نرمال شده و بین 0 و 1 قرار دارند.

numerical\_df.describe().show()



## تشخیص و حذف داده های پرت

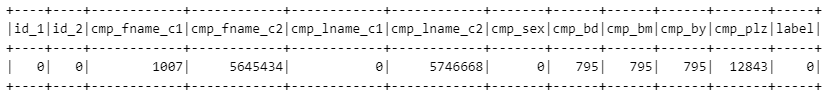
با توجه به اینکه ویژگی های ورودی همه نرمال شده هستند لذا هیچ داده پرتی هم وجود ندارد.

## بررسی مقادیر از دست رفته (**Missing Values**)

ابتدا توسط کد زیر وضعیت مقادیر از دست رفته را در مجموعه داده بررسی می کنیم:

numerical\_df.select([count(when(isnull(column), column)).alias(column) for column in numerical\_df.columns]).show()

نتیجه آن به صورت زیر است:



در ابتدا با توجه به وجود مقادیر از دست رفته در دو متغیر نمره گواهینامه و میانگین مسافت پیموده شده سالانه، تمامی رکوردهایی که دارای مقادیر از دست رفته هستند را حذف کرده و پس از ایجاد مقادیر از دست رفته مصنوعی و پر کردن آن‌ها با روش های مختلف و سنجیندن آن‌ها و انتخاب یکی از روش‌ها، رکوردهای حذف شده را برگردانده و با استفاده از روش انتخاب شده این مقادیر را پر می‌کنیم.

### ایجاد مقادیر از دست رفته مصنوعی (سلامی)

با فرض اینکه می‌خواهیم در تمامی متغیرها ۱۰ درصد مقادیر از دسته رفته داشته باشیم، تعداد آن‌ها را محاسبه کرده و به تعداد این مقادیر با استفاده از تابع رندوم اندیس‌هایی را تولید کرده و مقدار آن‌‌ها در متغیر مورد نظر را میسینگ می‌کنیم، همجنین مقدار اصلی را برای بررسی میزان مناسب بودن تابع‌های پر کردن میسینگ نگه‌داری می‌کنیم. قطعه کد استفاده شده به صورت زیر است:

outlier\_values\_missing = []

def create\_missing\_value(data):

df\_with\_missing = data.copy()

MISSING\_PERCENT = 0.1

for col in df\_with\_missing.columns:

if col == 'OUTCOME':

break

existing\_missing\_count = df\_with\_missing.isnull().sum()[col]

for i in range(int(MISSING\_PERCENT \* len(df\_with\_missing)) - existing\_missing\_count):

index = np.random.randint(0, high=len(df\_with\_missing))

while(pd.isnull(df\_with\_missing.loc[index, col])):

index = np.random.randint(0, high=len(data))

outlier\_values\_missing.append((col, index, df\_with\_missing[col][index]))

df\_with\_missing[col][index] = None

return df\_with\_missing

### پر کردن مقادیر از دست رفته و بررسی آن‌ها (رفیعی فرد-سلامی)

مقادیر از دست رفته ساخته شده را با استفاده دو روش KNN و Iterative پر کردیم. و در ادامه با استفاده از RMS آن‌ها را می‌سنجیم.

کد استفاده شده برای روش KNN به صورت زیر است:

def impute\_with\_knn(df, columns, n\_neighbors):

imputer = KNNImputer(n\_neighbors=n\_neighbors, weights='uniform', metric='nan\_euclidean')

imputer.fit(df)

Xtrans = imputer.transform(df)

return pd.DataFrame(Xtrans, columns=columns)

همچنین از قطعه کد زیر برای روش iterative استفاده کردیم:

def impute\_with\_iterative(df, columns, n\_neighbors):

imputer = IterativeImputer(max\_iter=10, random\_state=RANDOM\_STATE)

imputer.fit(df)

Xtrans = imputer.transform(df)

return pd.DataFrame(Xtrans, columns=columns)

در ادامه پس از پر کردن دیتا ها با روش ‌‌های بالا، از RMS استفاده کرده‌ایم تا بتوانیم دقت روش‌های بالا برای پر کردن دیتاهای میس شده را بررسی کنیم و بهترین روش ممکن را انتخاب کنیم.

def get\_column\_indexes(l):

return map(lambda tup: tup[1], list(filter(lambda tup: tup[0] == col, l)))

actual = []

predicted\_knn = []

predicted\_iterative = []

rms\_knn = []

rms\_iterative = []

for col in numerical\_df:

if col == 'ID':

continue

column\_indexes = get\_column\_indexes(outlier\_values\_missing)

for i in column\_indexes:

actual.append(outlier\_values\_missing[i][2])

predicted\_knn.append(knn\_filled\_df[col][i])

predicted\_iterative.append(iterative\_filled\_df[col][i])

rms\_knn.append((col, math.sqrt(metrics.mean\_squared\_error(actual, predicted\_knn))))

rms\_iterative.append((col, math.sqrt(metrics.mean\_squared\_error(actual, predicted\_iterative))))

## انتخاب متغیر‌ها (سلامی)

با توجه به اینکه تعداد متغیرهای موجود نسبتا زیاد است، نیاز داریم که با استفاده از روش‌های مختلف متغیرهای مناسبی که دارای همبسنگی بیشتری با متغیر هدف دارند را انتخاب کرده و آن‌ها را بررسی کنیم. این کار را با استفاده از دو روش random forest و boruta انجام می‌دهیم.

### انتخاب متغیر با random forest (سلامی)

برای انجام انتخاب متغیر‌ها از قطعه کد زیر استفاده می‌کنیم که درصد اهمیت متغیرها را محاسبه کرده و آن‌هایی را که از آستانه‌ی مورد نظر بالاتر است را انتخاب می‌کند.

def feature\_selection\_with\_RF(df, threshold):

x = df.drop("OUTCOME",axis=1)

y = df['OUTCOME']

x\_train, x\_test, y\_train, y\_test = train\_test\_split( x, y, test\_size=0.33, random\_state=RANDOM\_STATE)

rf = RandomForestRegressor(random\_state=RANDOM\_STATE)

rf.fit(x\_train, y\_train)

features = x.columns

f\_i = list(zip(features,rf.feature\_importances\_))

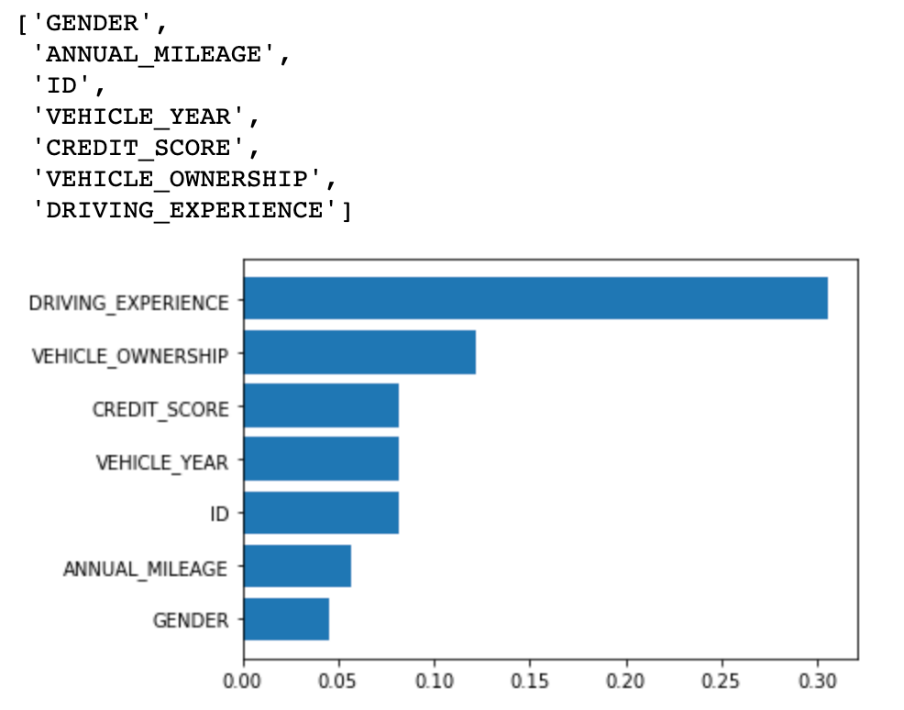
f\_i = list(filter(lambda x: x[1] > threshold, f\_i))

f\_i.sort(key = lambda x : x[1])

plt.barh([x[0] for x in f\_i],[x[1] for x in f\_i])

return list(map(lambda x: x[0], f\_i))

نتیجه‌ی کد بالا با آستانه‌ی 0.03 به صورت زیر است.



### انتخاب متغیر با Boruta (رفیعی فرد)

این روش با گرفتن متغیر‌های ورودی و خروجی تعدادی از متغیر‌های مناسب را انتخاب کرده و در خروجی نمایش می‌دهد:

def feature\_selection\_with\_BORUTA(X, Y):

X\_boruta = X.values

y\_boruta = Y.values

rf = RandomForestClassifier(n\_jobs=-1, n\_estimators=500, oob\_score=True, max\_depth=6)

feat\_selector = BorutaPy(rf, n\_estimators='auto', verbose=2, random\_state=RANDOM\_STATE)

feat\_selector.fit(X\_boruta, y\_boruta)

accept = X.columns[feat\_selector.support\_].to\_list()

irresolution = X.columns[feat\_selector.support\_weak\_].to\_list()

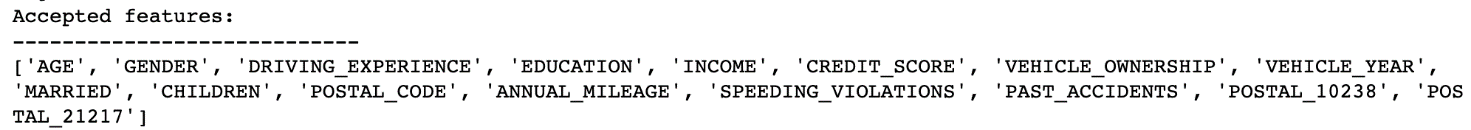
X\_filtered = feat\_selector.transform(X\_boruta)

print('Accepted features:')

print('----------------------------')

print(list(accept))

که نتیجه‌ی آن انتخاب متغیرهای زیر است:

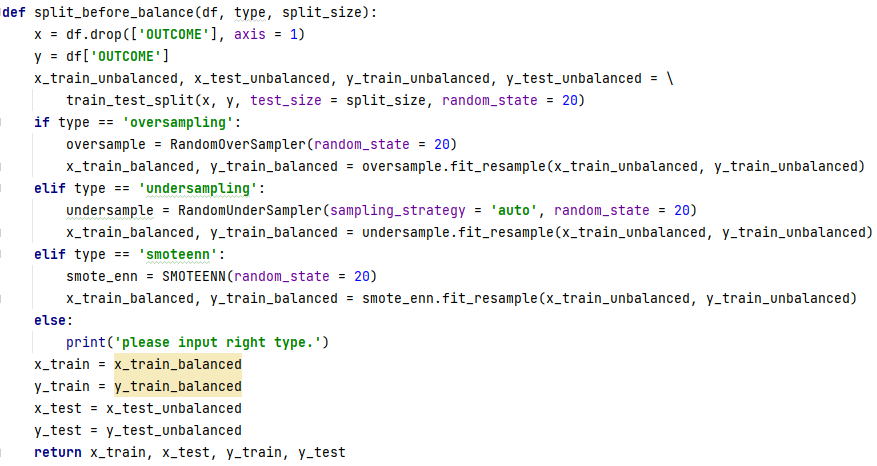


# ساخت مدل (رفیعی فرد)

پس از اتمام مراحل پیش پردازش با یک مجموعه داده عددی نرمال شده بدون مقادیر از دست رفته روبرو هستیم که می توانیم مراحل ساخت مدل را شروع کنیم. همانطور که در بخش 1 توضیح دادیم این مجموعه داده نامتقارن است.

## آماده سازی داده آموزشی و تست (رفیعی فرد)

ابتدا لازم بود داده تست و آموزشی از هم جدا بشود که با نسبت 70% داده آموزشی و 30% داده تست انجام شد. تابع زیر وظیفه انجام این کار را بر عهده دارد. در این تابع ابتدا متغیر خروجی حذف می شود سپس داده آموزشی از داده تست جدا می شود و در ادامه با توجه به اینکه متغیر خروجی نامتقارن است جهت جلوگیری از سوگیری نتایج به سمت کلاس اکثریت برای متقارن کردن آن سه روش بیش نمونه گیری، کم نمونه گیری و SMOTEENN در نظر گرفته شده است که بر حسب اینکه ورودی تابع چه مقداری باشد توسط یکی از روش های مذکور داده آموزشی را متقارن می کند.



با توجه به اینکه تعداد رکوردها حدود 10،000 عدد می باشد، لذا توسط دستور زیر با روش کم برازش داده ها متقارن گردید



در نتیجه اجرای دستور فوق تعداد رکوردهای داده آموزشی و تستبه ترتیب به شرح زیر می باشد:

Train:Counter({0.0: 2198, 1.0: 2198})

Test:Counter({0.0: 2072, 1.0: 921})

## معیار اصلی جهت انتخاب بهترین مدل (رفیعی فرد)

همانطور که در بخش های قبل ذکر شد مدل ساخته شده بر اساس مجموعه داده موجود باید برای یک مشتری جدید بر اساس اطلاعات ورودی او پیش بینی کند آیا برای اتومبیل خود درخواست خسارت می کند یا خیر.

بر این اساس مشتری پرخطر (زیان ده) اینگونه تعریف می گردد که در سال آتی از تصادف می کند و از شرکت بیمه درخواست خسارت خواهد کرد و مشتری کم خطر (سود ده) در سال آتی یا تصادف نمی کند و یا در صورت تصادف خسارتی از بیمه مطالبه نمی کند. به عبارتی دریافت یا عدم دریافت خسارت از بیمه مبنای تعیین مشتری پر خطر (زیان ده) و کم خطر (سود ده) خواهد بود.

لازم به یادآوری است عدد 0 در ستون نتیجه به معنای عدم مطالبه خسارت و عدد 1 به معنی مطالبه خسارت از طرف بیمه گذار می باشد.

بر اساس ماتریس آشفتگی که نتیجه اجرای مدل بر روی داده تست می باشد و در شکل زیر مشاهده می شود و با توجه به ماهیت مجموعه داده 4 حالت امکان رخ دادن وجود دارد:



1. TP : مدل به درستی پر خطر بودن بیمه گذار را تشخیص داده است به عبارتی خروجی مدل عدد 1 است و آنچه در ستون خروجی داده تست است هم عدد 1 است.
2. TN : مدل به درستی کم خطر بودن بیمه گذار را تشخیص داده است به عبارتی خروجی مدل عدد 0 و ستون خروجی آن بیمه گذار در داده تست نیز عدد 0 است.
3. FP : مدل به اشتباه بیمه گذار را یک راننده پرخطر شناسایی کرده است یعنی داده خروجی مدل عدد 1 است اما در داده تست برای او عدد 0 ذخیره شده است. به عبارت دیگر مدل یک مشتری سود ده را از دست داده است.
4. FN : مدل به اشتباه بیمه گذار را یک راننده کم خطر شناسایی کرده بطوریکه داده خروجی مدل برای این بیمه گذار عدد 0 است اما در مجموعه داده تست برای او عدد 1 وجود دارد. به عبارتی مدل یک مشتری زیان ده را جذب کرده است.

بر اساس صحبت با خبرگان صنعت بیمه و کسب نظرات آنها به این نتیجه گرفته شد که عدم جذب بیمه گذاران پر خطر (زیان ده) ارزش بالاتری نسبت به جذب بیمه گذاران کم خطر (سودآور) دارد. به عبارتی اگر به اشتباه یک راننده پر خطر با تصور اینکه این راننده درخواست مطالبه حق بیمه نخواهد کرد، جذب شود ضرر جبران ناپذیرتری به شرکت بیمه گر می زند تا اینکه یک راننده کم خطر با تصور اینکه این راننده درخواست مطالبه حق بیمه می کند، جذب نشود. در نتیجه گرچه FP باید مقدار کمی داشته باشد اما پایین بودن مقدار FN نسبت به FP اهمیت بیشتری دارد.

بر این اساس و با توجه به روابط فوق، معیار Recall نسبت به Precision در ارزیابی عملکرد دسته بند معیار مهمتری است ضمن اینکه بالا بودن بقیه معیارها هم باید در نظر گرفته شود.

با توجه به مطلب فوق معیار اصلی را برای ارزیابی دسته بندهاب مختلف یا هایپرپارامترهای مختلف یک دسته بند معیار Recall قرار می دهیم.

لازم به ذکر است که با توجه به داشتن دو کلاس در اینجا مقصود از معیار Recall همان معیار Recall برای کلاس 1 می باشد نه متوسط آن و نه برای کلاس 0 که مقدار آن بر اساس ماتریس آشفتگی از رابطه زیر به دست می آید:

## ساخت مدل با درخت تصمیم و تنظیم هایپرپارامترهای آن (رفیعی فرد)

متد ساخت مدل با این دسته بند به صورت زیر است:

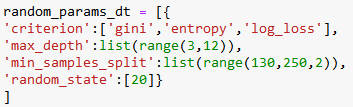
classifier = DecisionTreeClassifier(criterion, max\_depth, min\_samples\_split, class\_weight, random\_state)

### معرفی هایپرپارمترها (رفیعی فرد)

1. criterion معیار اندازه گیری خطا
2. max\_depth حداکثر عمق
3. min\_samples\_split حداقل تعداد داده در هر گره

### تنظیم پارامترها برای به دست آوردن بهترین حالت بر حسب معیار Recall (رفیعی فرد)

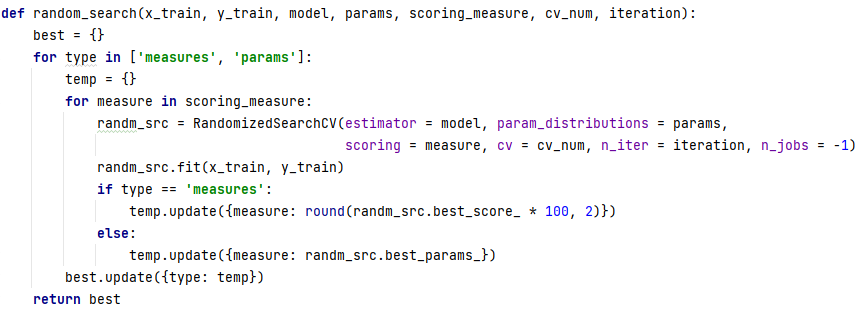
در ادامه بازه تغییر هایپرپارامترها برای استفاده از متد RandomizedSearchCV مشخص شد:



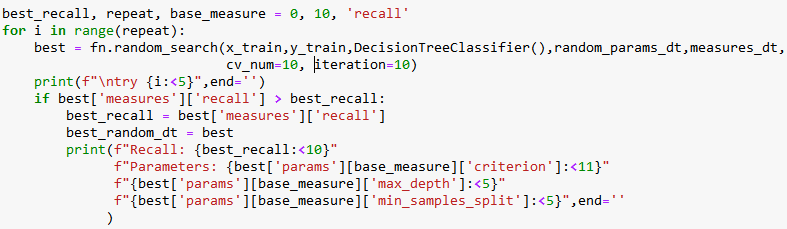
ابتدا توسط دستور زیر معیارهای مورد نظر برای ارزیابی مشخص شد:

measures\_dt = [**'recall'**,**'accuracy'**]

سپس تابع زیر جهت یافتن حالت بهینه بر اساس معیار مورد نظر استفاده شد:



در ادامه با استفاده از کد زیر با 10 بار اجرای تابع فوق بهترین حالت بر حسب معیار recall و accuracy به دست آمد:



نتایج به شرح زیر می باشد:

**try** 0 Recall: 84.39 Parameters: log\_loss 11 214   
**try** 1 Recall: 85.3 Parameters: gini 6 214   
**try** 2 Recall: 85.35 Parameters: gini 11 214   
**try** 3   
**try** 4   
**try** 5   
**try** 6   
**try** 7   
**try** 8   
**try** 9

**'measures'**: {**'recall'**: 85.35, **'accuracy'**: 84.01},  
**'params'**: {

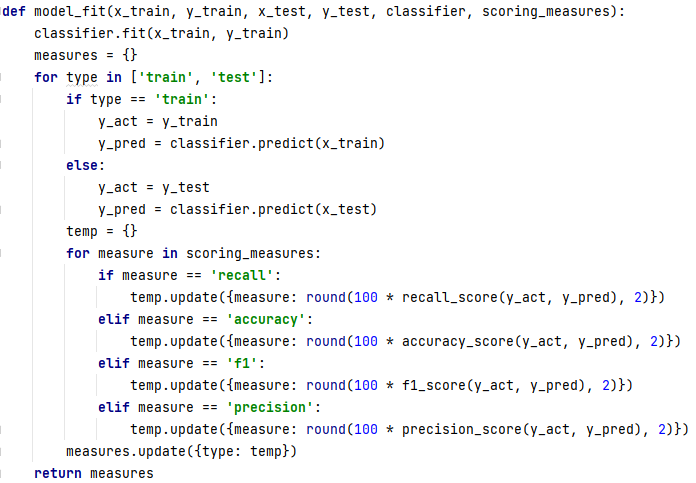
**'recall'**:

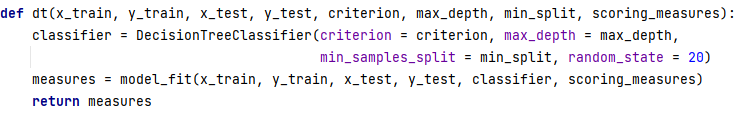
{**'min\_samples\_split'**: 214,  
 **'max\_depth'**: 11,  
 **'criterion'**: **'gini'**},  
**'accuracy'**:

{**'min\_samples\_split'**: 144,  
 **'max\_depth'**: 7,  
 **'criterion'**: **'** **entropy'**}}

### ارزیابی مدل بر روی داده آموزشی و تست (رفیعی فرد)

دو تابع زیر برای ارزیابی مدل استفاده شد:



****

سپس با کد زیر مقدار معیار recall و accuracy برای داده آموزشی و تست محاسبه شد:

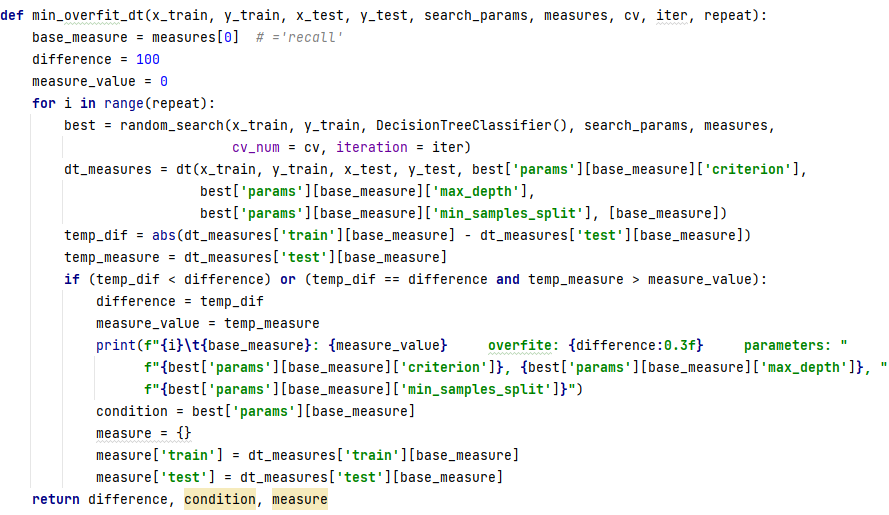


نتایج به صورت زیر به دست آمد:

**'train'**: {**'recall'**: 84.39, **'accuracy'**: 84.28},  
 **'test'**: {**'recall'**: 84.8, **'accuracy'**: 83.83}

### جستجوی کمترین حالت overfitting (رفیعی فرد)

تابع زیر برای یافتن کمترین اختلاف معیار Recall بین داده آموزشی و تست استفاده شد:



سپس با استفاده از کد زیر، حالتی را که کمترین overfitting اتفاق بیفتد به دست آمد:



نتایج به شرح زیر است:

try 0 recall: 85.88 overfite: 0.580 parameters: gini, 3, 202  
try 1 recall: 82.63 overfite: 0.460 parameters: log\_loss, 9, 192  
try 2 recall: 82.74 overfite: 0.390 parameters: gini, 6, 196  
try 9 recall: 84.04 overfite: 0.350 parameters: log\_loss, 5, 176  
try 15 recall: 84.58 overfite: 0.180 parameters: log\_loss, 3, 240

در هر بار سعی، اگر نتایج بهتر شده باشد آن را چاپ می کند. بهترین نتیجه هایلایت شده است. مقادیر Recall روی داده تست هستند. ملاحظه می شود که فقط 0.18 دچار overfitting شده است.

با استفاده از دستور زیر نتایج برای داده آموزشی و تست بر اساس دو معیار نمایش داده می شود

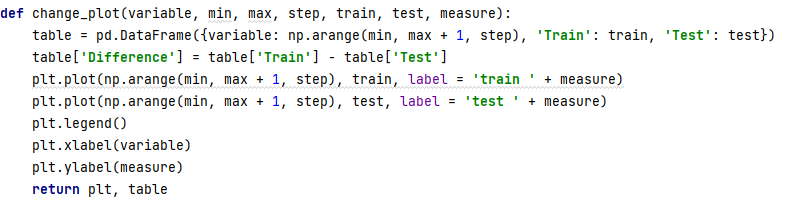


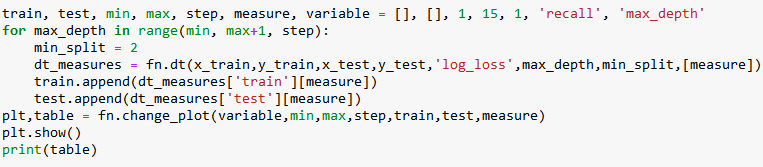
نتایج به شرح زیر است:

**'train'**: {**'recall'**: 84.76, **'accuracy'**: 82.83}  
**'test'**: {**'recall'**: 84.58, **'accuracy'**: 81.72}

### بررسی تغییر نتایج با تغییر هایپرپارامترها (رفیعی فرد)

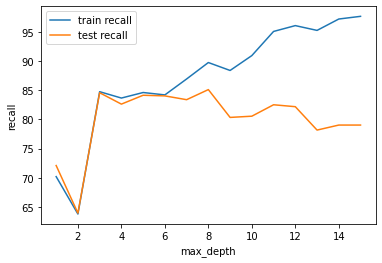
با استفاده از کد زیر نمودار تغییرات Recall برای داده آموزشی و تست بر حسب عمق رسم شد:





نکته ای که لازم است در مورد کد بالا به آن اشاره شود این است که جهت یافتن عمق بهینه و برای اینکه حداقل سایز داده ها مانعی برای این موضوع نباشد min\_split\_size برابر با کمترین مقدار ممکن یعنی 2 قرار داده شد.

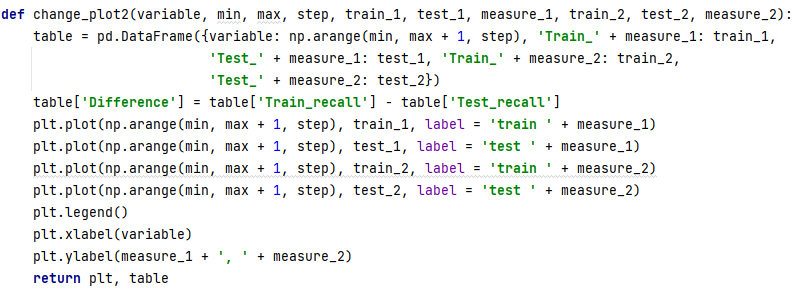
خروجی کد بالا به صورت نمودار و داده جدولی به صورت زیر است:

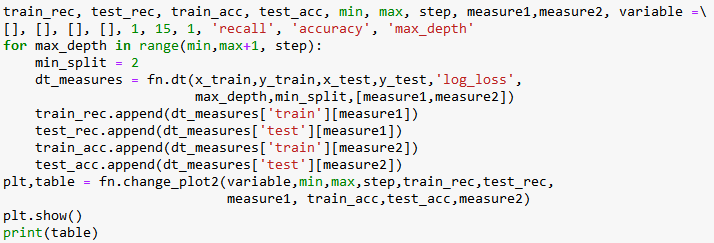


|  |  |  |  |
| --- | --- | --- | --- |
| max\_depth | Train | Test | Difference |
| 1 | 70.20 | 72.10 | -1.90 |
| 2 | 63.79 | 63.95 | -0.16 |
| 3 | 84.76 | 84.58 | 0.18 |
| 4 | 83.67 | 82.63 | 1.04 |
| 5 | 84.62 | 84.15 | 0.47 |
| 6 | 84.21 | 84.04 | 0.17 |
| 7 | 86.94 | 83.39 | 3.55 |
| 8 | 89.76 | 85.12 | 4.64 |

همانگونه که ملاحظه می شود تا عمق 6 نتایج بر روی داده آموزشی و تست رو به بهبود هستند اما با افزایش عمق از این نقطه نتایج برای داده آموزشی رو به افزایش و برای داده تست رو به کاهش می گذارد و با افزایش عمق مدل به سمت overfitting بیشتر می رود. بین عمق 3 و عمق 6 نتایج خوب است با اینحال در عمق 3 هم مقدار overfitting قابل قبول است هم نتایج روی داده تست کمی بهتر از عمق 6 است لذا عمق 3 نقطه بهینه است. جهت خوانایی مطالب مقادیر در جدول تا عمق 8 آورده شده است.

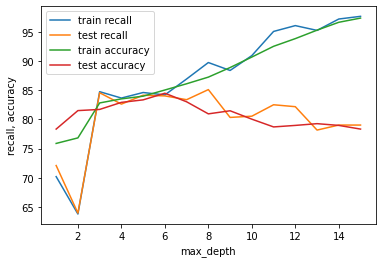
با استفاده از کد زیر نمودار Recall و Accuracy بر حسب عمق رسم شد:

****



نتایج به صورت زیر است:

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| max\_depth | Train\_recall | Test\_recall | Train\_accuracy | Test\_accuracy | Difference |
| 1 | 70.20 | 72.10 | 75.89 | 78.35 | -1.90 |
| 2 | 63.79 | 63.95 | 76.84 | 81.52 | -0.16 |
| 3 | 84.76 | 84.58 | 82.83 | 81.72 | 0.18 |
| 4 | 83.67 | 82.63 | 83.53 | 82.93 | 1.04 |
| 5 | 84.62 | 84.15 | 83.99 | 83.36 | 0.47 |
| 6 | 84.21 | 84.04 | 85.05 | 84.50 | 0.17 |
| 7 | 86.94 | 83.39 | 86.12 | 83.03 | 3.55 |
| 8 | 89.76 | 85.12 | 87.26 | 80.96 | 4.64 |
| 9 | 88.40 | 80.35 | 88.90 | 81.49 | 8.05 |
| 10 | 90.95 | 80.56 | 90.70 | 80.05 | 10.39 |
| 11 | 95.09 | 82.52 | 92.54 | 78.72 | 12.57 |
| 12 | 96.09 | 82.19 | 93.86 | 78.98 | 13.90 |
| 13 | 95.27 | 78.18 | 95.31 | 79.28 | 17.09 |
| 14 | 97.22 | 79.04 | 96.66 | 78.98 | 18.18 |
| 15 | 97.68 | 79.04 | 97.38 | 78.35 | 18.64 |



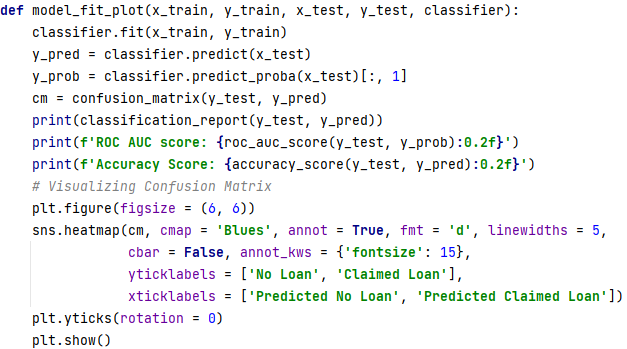
### بهترین هایپرپارامترها و رسم نمودار نتایج بهینه (رفیعی فرد)

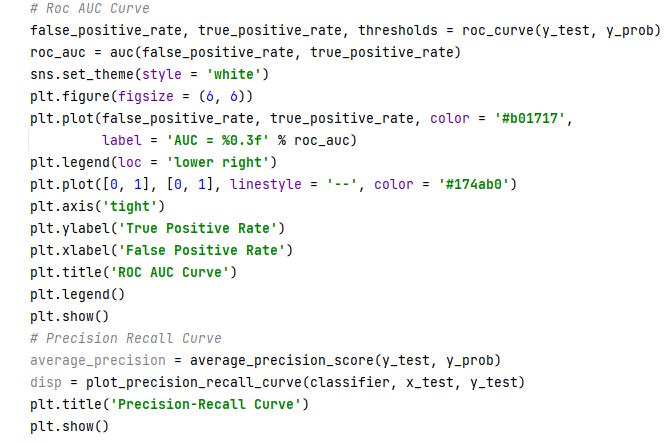
با توجه به نتایج حاصل از جستجوی تصادفی در فضای هایپرپارامترها و نمودارهای بالا بهترین هایپرپارامترها به شرح زیر هستند:

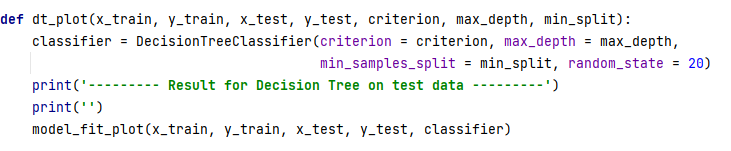
**'min\_samples\_split'**: 2,  
**'max\_depth'**: 3,  
**'criterion'**: **'log\_loss'**

لازم به ذکر است با توجه به اینکه دو هایپرپارامتر **min\_samples\_split** و **max\_depth** به هم وابستگی دارند و تعیین یکی از آنها بر دیگری تأثیر می گذارد بنابراین اینکه مقدار **max\_depth** برابر با 2 باشد یا 240 تاثیری در نتایج نمی گذارد چون پارامتر محدود کننده، **min\_samples\_split** است.

جهت رسم ماتریس آشفتگی، نمودار ROC AUC و نمودار Precision بر حسب Recall از کد زیر استفاده شده است:

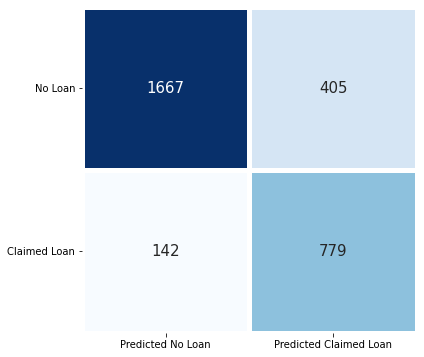
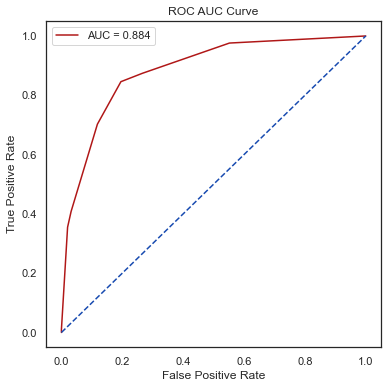


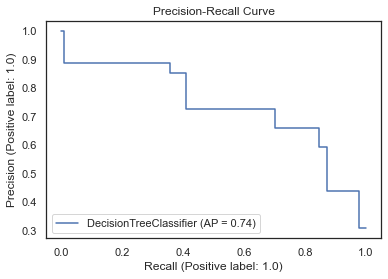




نتایج به صورت زیر هستند:

--------- Result **for** Decision Tree on test data ---------  
  
 precision recall f1-score support  
  
 0.0 0.92 0.80 0.86 2072  
 1.0 **0.66** **0.85** **0.74** 921  
  
 accuracy **0.82** 2993  
 macro avg 0.79 0.83 0.80 2993  
weighted avg 0.84 0.82 0.82 2993  
  
ROC AUC score: **0.88**  
Accuracy Score: 0.82



## ساخت مدل با **Bagging** و مدل پایه درخت تصمیم و تنظیم هایپرپارامترهای آن (رفیعی فرد)

متد ساخت مدل:

classifier = BaggingClassifier(base\_estimator, max\_features, max\_samples,

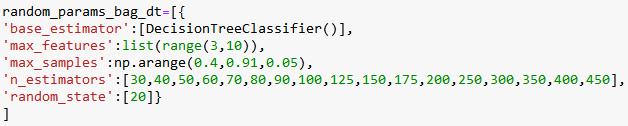
n\_estimators, random\_state=20)

### معرفی هایپرپارمترها (رفیعی فرد)

1. base\_estimator مدل پایه ای که بر اساس آن مدل سازی انجام می شود. در این قسمت درخت تصمیم است
2. max\_features حداکثر تعداد ویژگی هایی که در هر بار ساخت مدل با دسته بند پایه (در اینجا درخت تصمیم) استفاده می شود
3. max\_samples حداکثر تعداد نمونه هایی که در در هر بار ساخت مدل با دسته بند پایه (در اینجا درخت تصمیم) استفاده می شود
4. n\_estimators تعداد دفعات ساخت مدل با دسته بند پایه (در اینجا درخت تصمیم)

### تنظیم پارامترها برای به دست آوردن بهترین حالت بر حسب معیار Recall (رفیعی فرد)

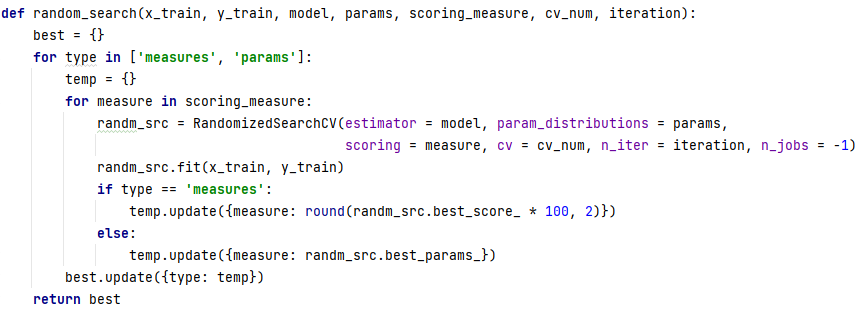
در ادامه بازه تغییر هایپرپارامترها برای استفاده از متد RandomizedSearchCV مشخص شد:



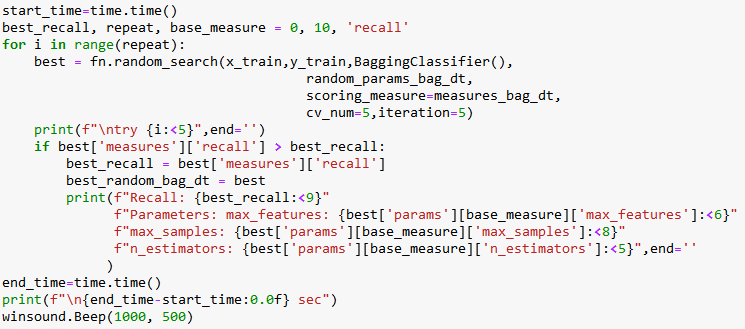
به این دلیل که زمان اجرای این دسته بند طولانی است، برای جلوگیری از بالا رفتن زمان اجرا فقط معیار Recall محاسبه گردید. با کد زیر معیار مورد نظر را مشخص می کنیم:

measures\_bag\_dt = [**'recall'**]

ابتدا تابع زیر جهت یافتن حالت بهینه بر اساس معیار مورد نظر استفاده شد:



سپس با استفاده از کد زیر با 10 بار اجرای تابع فوق بهترین حالت بر حسب معیار recall به دست آمد:



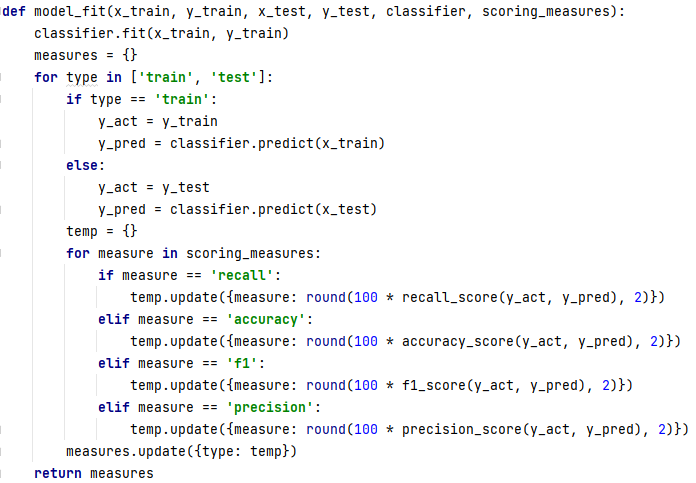
نتایج به شرح زیر می باشد:

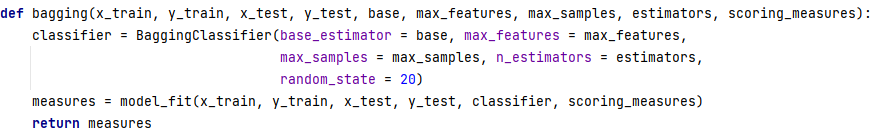
**try** 0 Recall: 83.21 Parameters: max\_features: 6 max\_samples: 0.7 n\_estimators: 60  
**try** 1 Recall: 83.35 Parameters: max\_features: 6 max\_samples: 0.85 n\_estimators: 100  
**try** 2   
**try** 3 Recall: 83.39 Parameters: max\_features: 6 max\_samples: 0.65 n\_estimators: 60  
**try** 4   
**try** 5   
**try** 6   
**try** 7   
**try** 8   
**try** 9   
641 sec

**'measures'**: {**'recall'**: 83.39}  
**'params'**:  
 **'n\_estimators'**: 60  
 **'max\_samples'**: 0.65  
 **'max\_features'**: 6

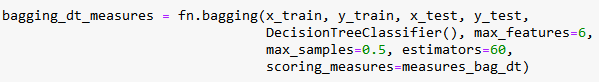
### ارزیابی مدل بر روی داده آموزشی و تست (رفیعی فرد)

دو تابع زیر برای ارزیابی مدل استفاده شد:



****

سپس با کد زیر مقدار معیار recall برای داده آموزشی و تست محاسبه شد:

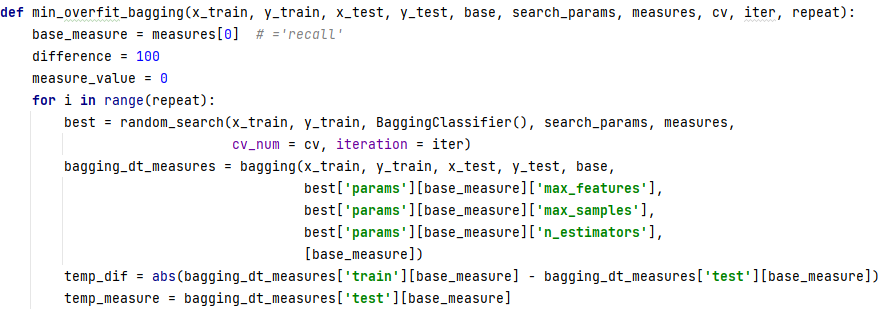


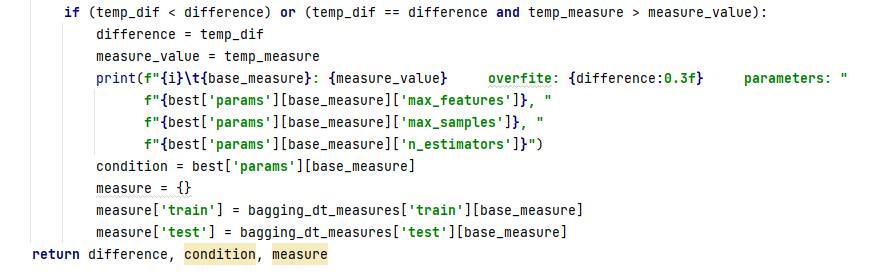
نتایج به صورت زیر به دست آمد:

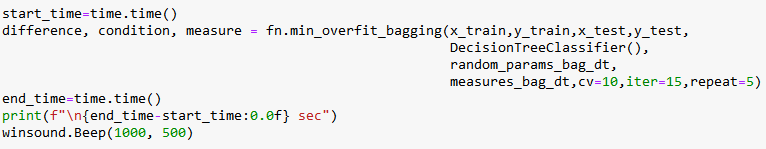
**'train'**: {**'recall'**: 91.81}, **'test'**: {**'recall'**: 83.82}

### جستجوی کمترین حالت overfitting (رفیعی فرد)

تابع زیر برای یافتن کمترین اختلاف معیار Recall بین داده آموزشی و تست استفاده شد:



 سپس با استفاده از کد زیر، حالتی را که کمترین overfitting اتفاق بیفتد به دست آمد:

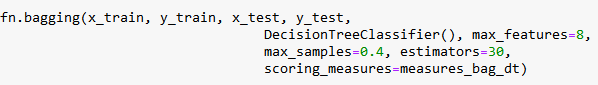


نتایج به شرح زیر است:

try 0 recall: 84.47 overfite: 7.610 parameters: max\_features: 8, max\_samples: 0.40, n\_estimators: 30  
  
705 sec

در هر بار سعی، اگر نتایج بهتر شده باشد آن را چاپ می کند. مقادیر Recall روی داده تست هستند. ملاحظه می شود که در بهترین حالت دارای 7.6% overfitting است.

با استفاده از دستور زیر نتایج برای داده آموزشی و تست بر اساس معیار Recall نمایش داده می شود:



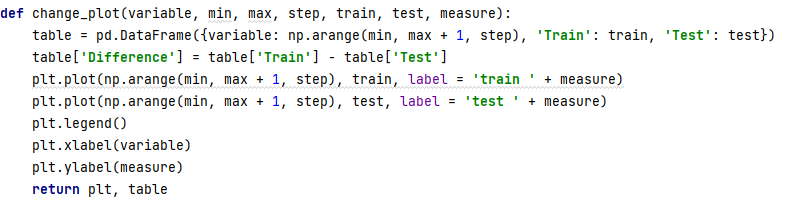
نتایج به شرح زیر است:

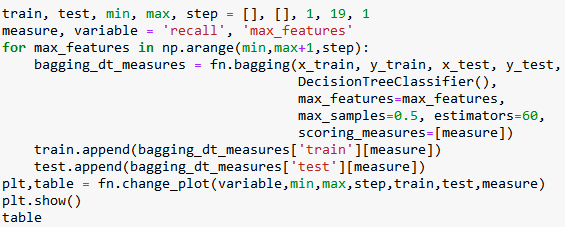
train**': {'**recall**': 92.08}, '**test**': {'**recall**': 84.47}**

لازم به توضیح است که هر چقدر هایپر پارامترها تغییر داده شد خیلی موفقیت در کم شدن overfitting به دست نیامد و پس از تغییرات مکرر بر روی هایپرپارامترها نهایتاً مقدار overfitting از حدود 11 به 7.6 کاهش پیدا کرد. یکی از مواردی که تأثیر زیادی در overfitting مدل می گذارد پارامتر n\_stimator است که با تقلیل این پارامتر موفقیت نسبی در کم کردن overfitting حاصل شد. با اینحال با توجه به نوع داده های به نظر می آید این مدل با این دسته بند، مدل مناسبی برای این داده ها نباشد.

### بررسی تغییر نتایج با تغییر هایپرپارامترها (رفیعی فرد)

با استفاده از کد زیر نمودار تغییرات Recall برای داده آموزشی و تست بر حسب max\_features رسم شد:





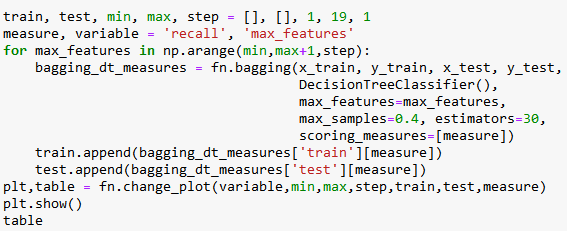
خروجی کد بالا به صورت نمودار و داده جدولی به صورت زیر است:



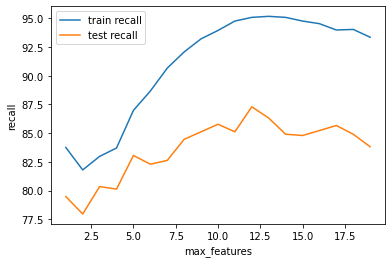
|  |  |  |  |
| --- | --- | --- | --- |
| max\_features | Train | Test | Difference |
| 1 | 84.08 | 83.60 | 0.48 |
| 2 | 84.85 | 80.46 | 4.39 |
| 3 | 86.03 | 80.78 | 5.25 |
| 4 | 87.12 | 81.65 | 5.47 |
| 5 | 89.40 | 83.06 | 6.34 |
| 6 | 91.81 | 83.82 | 7.99 |
| 7 | 93.22 | 83.82 | 9.40 |
| 8 | 94.27 | 84.91 | 9.36 |
| 9 | 96.13 | 85.34 | 10.79 |
| 10 | 96.18 | 86.32 | 9.86 |
| 11 | 96.95 | 86.32 | 10.63 |
| 12 | 97.86 | 86.75 | 11.11 |
| 13 | 97.91 | 85.34 | 12.57 |
| 14 | 97.91 | 85.23 | 12.68 |
| 15 | 97.95 | 85.23 | 12.72 |
| 16 | 97.09 | 84.26 | 12.83 |
| 17 | 97.41 | 85.45 | 11.96 |
| 18 | 97.13 | 84.58 | 12.55 |
| 19 | 96.91 | 84.15 | 12.76 |

ملاحظه می شود که خروجی مدل برای داده تست تا max\_feature=12 افزایش داشته و از آنجا به بعد رو به کاهش می گذارد.

مجدداً همین نمودار با هایپرپارامترهای خروجی تابع کمترین overfitting، رسم شد که کد آن به صورت زیر می باشد:

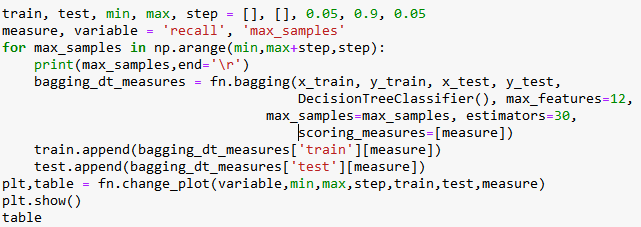


نتایح رسم نمودار و داده جدولی به صورت زیر است:

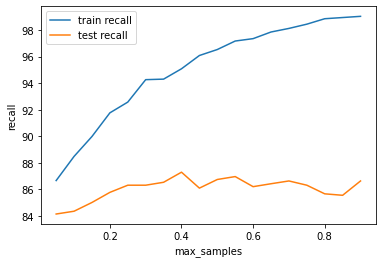


|  |  |  |  |
| --- | --- | --- | --- |
| max\_features | Train | Test | Difference |
| 1 | 83.76 | 79.48 | 4.28 |
| 2 | 81.80 | 77.96 | 3.84 |
| 3 | 82.98 | 80.35 | 2.63 |
| 4 | 83.71 | 80.13 | 3.58 |
| 5 | 86.99 | 83.06 | 3.93 |
| 6 | 88.67 | 82.30 | 6.37 |
| 7 | 90.67 | 82.63 | 8.04 |
| 8 | 92.08 | 84.47 | 7.61 |
| 9 | 93.22 | 85.12 | 8.10 |
| 10 | 93.95 | 85.78 | 8.17 |
| 11 | 94.77 | 85.12 | 9.65 |
| 12 | 95.09 | 87.30 | 7.79 |
| 13 | 95.18 | 86.32 | 8.86 |
| 14 | 95.09 | 84.91 | 10.18 |
| 15 | 94.77 | 84.80 | 9.97 |
| 16 | 94.54 | 85.23 | 9.31 |
| 17 | 93.99 | 85.67 | 8.32 |
| 18 | 94.04 | 84.91 | 9.13 |
| 19 | 93.36 | 83.82 | 9.54 |

با این هایپرپارامترها هم دقت مدل روی داده تست بالاتر رفت و هم overfitting کاهش پیدا کرد. البته نسبت به حالت کمترین overfitting اندکی overfitting بیشتری دارد که بسیار مقدار ناچیزی است. لذا با قرار دادن max\_features=12 تغییرات دقت مدل برای داده تست را برای هایپر پارامتر max\_samples بررسی کردیم. کد این قسمت به صورت زیر می باشد:

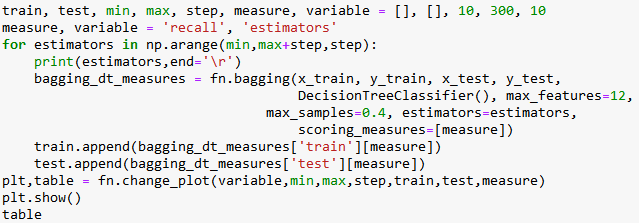


نتایج به صورت زیر می باشد:

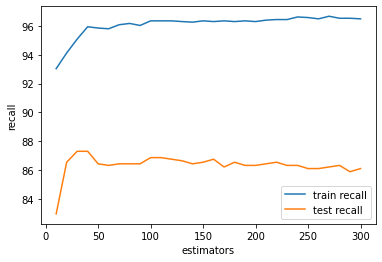


|  |  |  |  |
| --- | --- | --- | --- |
| max\_samples | Train | Test | Difference |
| 0.05 | 86.67 | 84.15 | 2.52 |
| 0.10 | 88.49 | 84.36 | 4.13 |
| 0.15 | 89.99 | 85.02 | 4.97 |
| 0.20 | 91.77 | 85.78 | 5.99 |
| 0.25 | 92.58 | 86.32 | 6.26 |
| 0.30 | 94.27 | 86.32 | 7.95 |
| 0.35 | 94.31 | 86.54 | 7.77 |
| 0.40 | 95.09 | 87.30 | 7.79 |
| 0.45 | 96.09 | 86.10 | 9.99 |
| 0.50 | 96.54 | 86.75 | 9.79 |
| 0.55 | 97.18 | 86.97 | 10.21 |
| 0.60 | 97.36 | 86.21 | 11.15 |
| 0.65 | 97.86 | 86.43 | 11.43 |
| 0.70 | 98.13 | 86.64 | 11.49 |
| 0.75 | 98.45 | 86.32 | 12.13 |
| 0.80 | 98.86 | 85.67 | 13.19 |
| 0.85 | 98.95 | 85.56 | 13.39 |
| 0.90 | 99.04 | 86.64 | 12.40 |

ملاحظه می شود که max\_samples=0.4 که در قسمت 4-4-3 به دست آمده بود اینجا هم مورد تأیید قرار می گیرد و بهترین حالت برای این هایپرپارامتر همین مقدار است. در مرحله بعد با قرار دادن max\_samples=0.4 و max\_features=12 تغییرات دقت مدل بر حسب n\_estimators بررسی شد. کد این قسمت به صورت زیر می باشد:

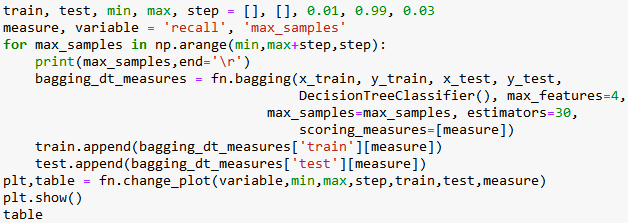


نتایج به صورت زیر می باشد:

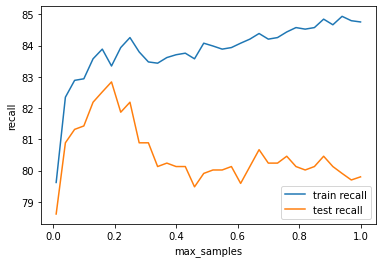


|  |  |  |  |
| --- | --- | --- | --- |
| estimators | Train | Test | Difference |
| 10 | 93.04 | 82.95 | 10.09 |
| 20 | 94.13 | 86.54 | 7.59 |
| 30 | 95.09 | 87.30 | 7.79 |
| 40 | 95.95 | 87.30 | 8.65 |
| 50 | 95.86 | 86.43 | 9.43 |
| 60 | 95.81 | 86.32 | 9.49 |
| 100 | 96.36 | 86.86 | 9.50 |
| 120 | 96.36 | 86.75 | 9.61 |
| 140 | 96.27 | 86.43 | 9.84 |
| 180 | 96.31 | 86.54 | 9.77 |
| 200 | 96.31 | 86.32 | 9.99 |
| 220 | 96.45 | 86.54 | 9.91 |
| 240 | 96.63 | 86.32 | 10.31 |

با انتخاب هایپرپارامترهای زیر و رسم نمودار Recall بر حسب max\_samples مدل تقریباً از حالت overfitting خارج شد که با استفاده از کد زیر نمودار ترسیم شد:

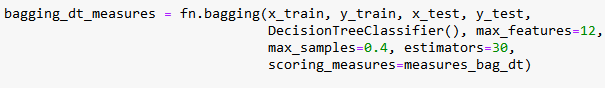


که نتایج به صورت زیر هستند:



|  |  |  |  |
| --- | --- | --- | --- |
| max\_samples | Train | Test | Difference |
| 0.01 | 79.62 | 78.61 | 1.01 |
| 0.07 | 82.89 | 81.32 | 1.57 |
| 0.10 | 82.94 | 81.43 | 1.51 |
| 0.13 | 83.58 | 82.19 | 1.39 |
| 0.16 | 83.89 | 82.52 | 1.37 |
| 0.19 | 83.35 | 82.84 | 0.51 |
| 0.22 | 83.94 | 81.87 | 2.07 |
| 0.25 | 84.26 | 82.19 | 2.07 |
| 0.31 | 83.48 | 80.89 | 2.59 |
| 0.43 | 83.76 | 80.13 | 3.63 |
| 0.49 | 84.08 | 79.91 | 4.17 |
| 0.55 | 83.89 | 80.02 | 3.87 |
| 0.61 | 84.08 | 79.59 | 4.49 |
| 0.73 | 84.26 | 80.24 | 4.02 |
| 0.79 | 84.58 | 80.13 | 4.45 |
| 0.85 | 84.58 | 80.13 | 4.45 |
| 0.91 | 84.67 | 80.13 | 4.54 |
| 0.97 | 84.80 | 79.70 | 5.10 |
| 1.00 | 84.76 | 79.80 | 4.96 |

با توجه به نمودار فوق برای max\_samples=0.19 کمترین overfitting وجود دارد ولی مقدار Recall حدود 5% کمتر از حالت بهینه است و از آنجاکه مقدار Recall بالاتر (برای داده تست) اهمیت بیشتری دارد لذا نتایج قبلی ارجحیت دارد. بنابراین با توجه به نمودارهای بالا و توسط کد زیر نتیجه بهینه به دست آمد:



که نتایج به شرح زیر هستند:

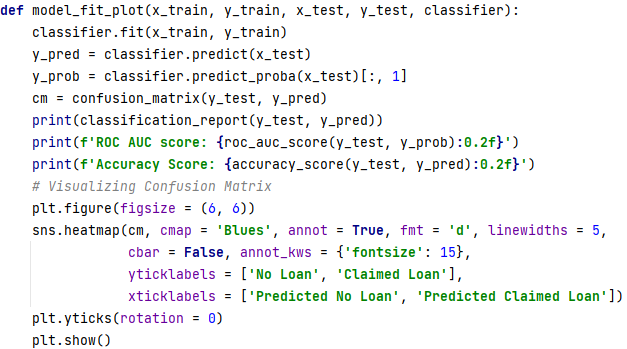
**'train'**: {**'recall'**: 95.09}, **'test'**: {**'recall'**: 87.3}

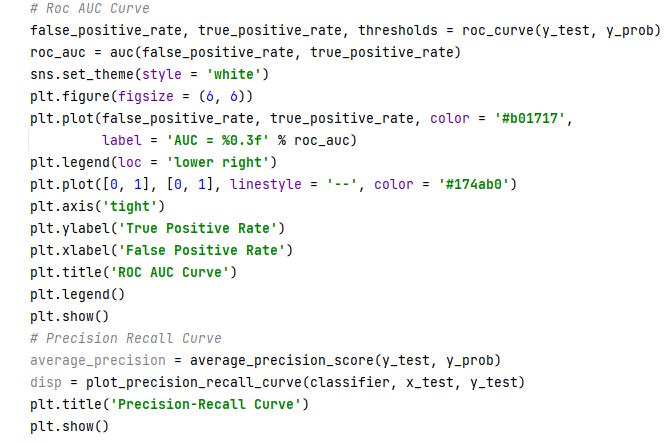
### بهترین هایپرپارامترها و رسم نمودار نتایج بهینه (رفیعی فرد)

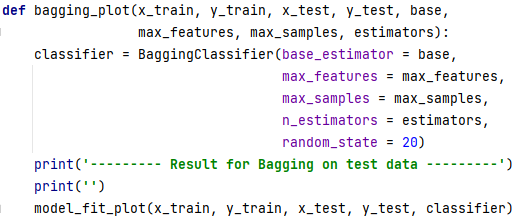
با توجه به نتایج حاصل از جستجوی تصادفی در فضای هایپرپارامترها و نمودارهای بالا بهترین هایپرپارامترها به شرح زیر هستند:

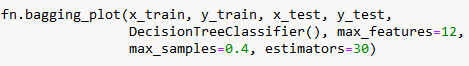
**max\_features**: 12  
**max\_samples**: 0.4  
**n\_estimators**: 30

جهت رسم ماتریس آشفتگی، نمودار ROC AUC و نمودار Precision بر حسب Recall از کد زیر استفاده شد:



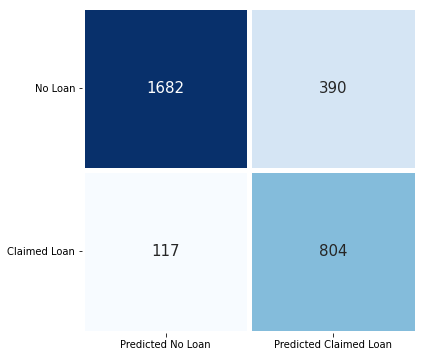
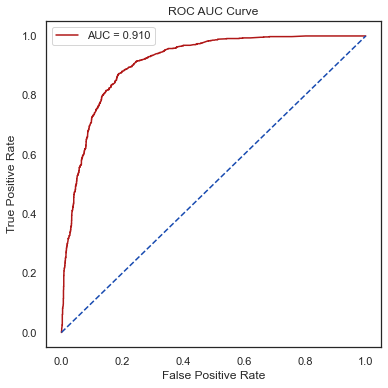


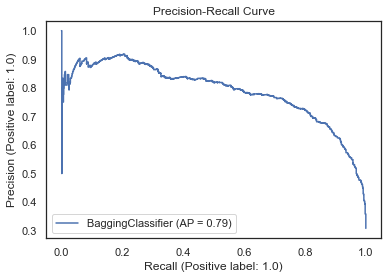




نتایج به صورت زیر هستند:

--------- Result **for** Bagging on test data ---------  
  
 precision recall f1-score support  
  
 0.0 0.93 0.81 0.87 2072  
 1.0 **0.67 0.87 0.76** 921  
  
 accuracy **0.83** 2993  
 macro avg 0.80 0.84 0.81 2993  
weighted avg 0.85 0.83 0.84 2993  
  
ROC AUC score: 0.91  
Accuracy Score: 0.83



## **ساخت مدل با Bagging و مدل پایه KNN و تنظیم هایپرپارامترها** (رفیعی فرد)

متد ساخت مدل:

classifier = BaggingClassifier(base\_estimator, max\_features, max\_samples,

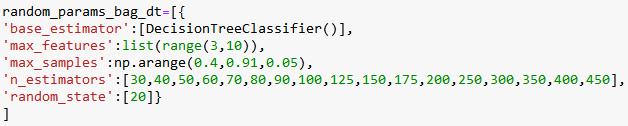
n\_estimators, random\_state=20)

### معرفی هایپرپارمترها (رفیعی فرد)

1. base\_estimator مدل پایه ای که بر اساس آن مدل سازی انجام می شود. در این قسمت KNN است
2. max\_features حداکثر تعداد ویژگی هایی که در هر بار ساخت مدل با دسته بند پایه (در اینجا KNN) استفاده می شود
3. max\_samples حداکثر تعداد نمونه هایی که در در هر بار ساخت مدل با دسته بند پایه (در اینجا KNN) استفاده می شود
4. n\_estimators تعداد دفعات ساخت مدل با دسته بند پایه (در اینجا KNN)

### تنظیم پارامترها برای به دست آوردن بهترین حالت بر حسب معیار Recall (رفیعی فرد)

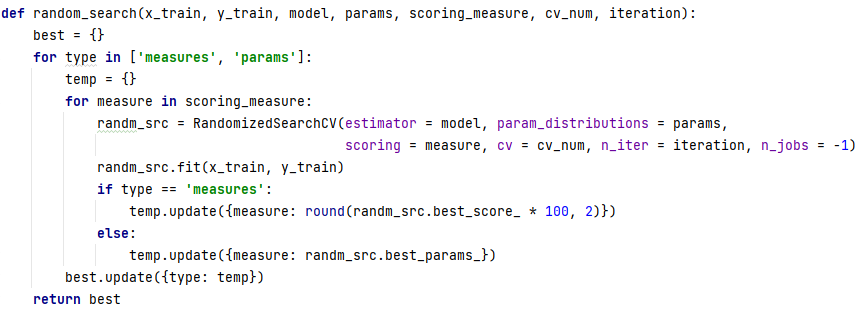
در ادامه بازه تغییر هایپرپارامترها برای استفاده از متد RandomizedSearchCV مشخص شد:



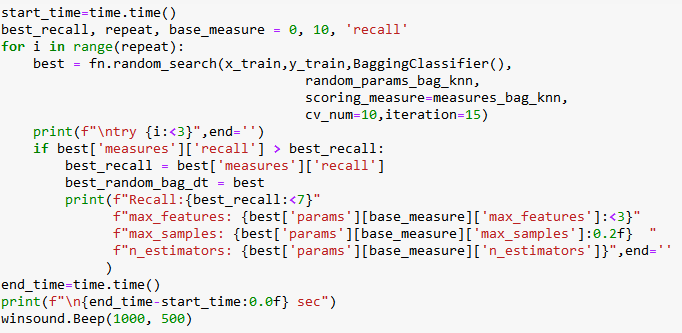
به این دلیل که زمان اجرای این دسته بند طولانی است، برای جلوگیری از بالا رفتن زمان اجرا فقط معیار Recall محاسبه گردید. با کد زیر معیار مورد نظر را مشخص می کنیم:

measures\_bag\_dt = [**'recall'**]

ابتدا تابع زیر جهت یافتن حالت بهینه بر اساس معیار مورد نظر استفاده شد:



سپس با استفاده از کد زیر با 10 بار اجرای تابع فوق بهترین حالت بر حسب معیار recall به دست آمد:



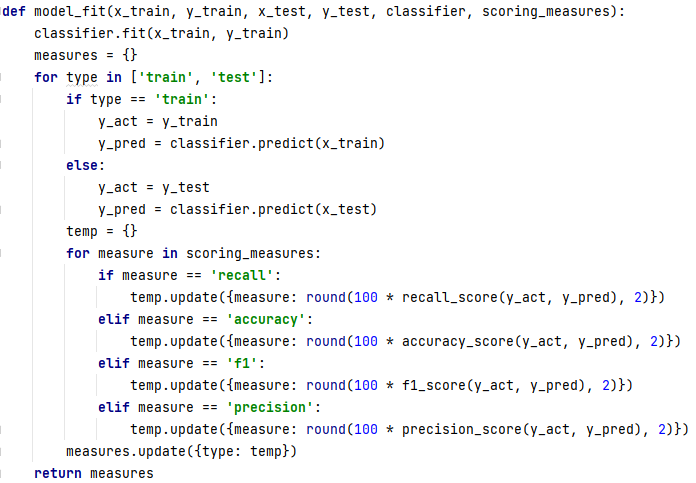
نتایج به شرح زیر می باشد:

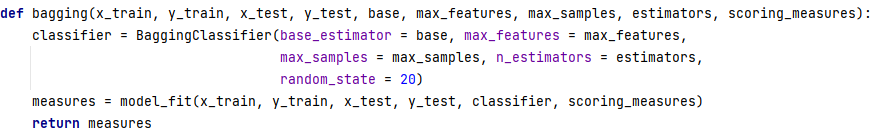
**try** 0 Recall:86.67 max\_features: 9 max\_samples: 0.50 n\_estimators: 200  
**try** 1   
**try** 2 Recall:87.17 max\_features: 9 max\_samples: 0.90 n\_estimators: 80  
**try** 3   
**try** 4   
**try** 5   
**try** 6   
**try** 7   
**try** 8   
**try** 9   
7939 sec

**'measures'**: {**'recall'**: 87.17}  
**'params'**:  
 **'n\_estimators'**: 80  
 **'max\_samples'**: 0.9  
 **'max\_features'**: 9

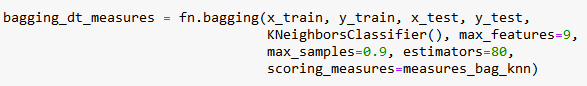
### ارزیابی مدل بر روی داده آموزشی و تست (رفیعی فرد)

دو تابع زیر برای ارزیابی مدل استفاده شد:



****

سپس با کد زیر مقدار معیار recall برای داده آموزشی و تست محاسبه شد:



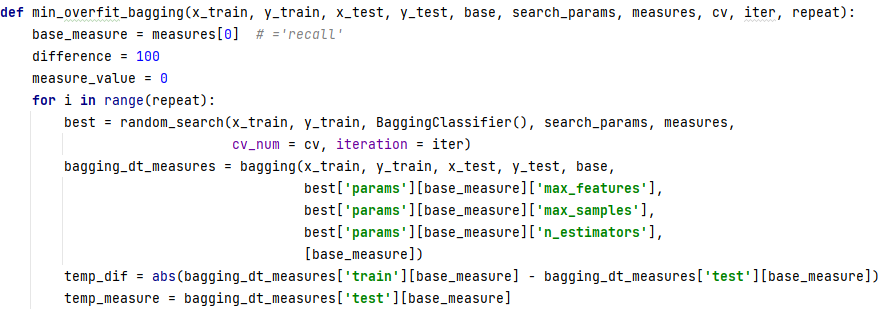
نتایج به صورت زیر به دست آمد:

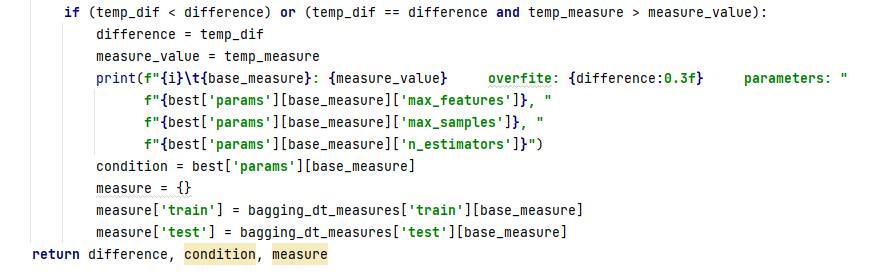
**'train'**: {**'recall'**: 92.17}, **'test'**: {**'recall'**: 86.43}

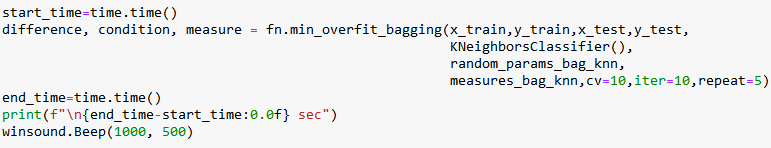
بر اساس این نتایج مدل تا حدی دچار overfitting شده است که در ادامه سعی شده است نتایج بهتری از این حیث حاصل شود.

### جستجوی کمترین حالت overfitting (رفیعی فرد)

تابع زیر برای یافتن کمترین اختلاف معیار Recall بین داده آموزشی و تست استفاده شد:



 سپس با استفاده از کد زیر، حالتی را که کمترین overfitting اتفاق بیفتد به دست آمد:

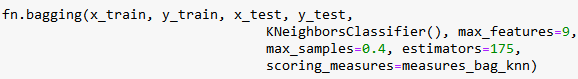


نتایج به شرح زیر است:

**try** 0 recall: 87.08 overfite: 2.910 max\_features: 9 max\_samples: 0.40 n\_estimators: 175  
  
1665 sec

در هر بار سعی، اگر نتایج بهتر شده باشد آن را چاپ می کند. مقادیر Recall روی داده تست هستند. ملاحظه می شود که در بهترین حالت دارای 2.91% overfitting است که در ادامه بررسی می شود که آیا می توان این مقدار را کاهش داد یا خیر.

با استفاده از دستور زیر نتایج برای داده آموزشی و تست بر اساس معیار Recall نمایش داده می شود:



نتایج به شرح زیر است:

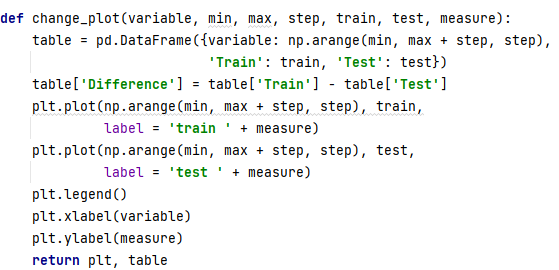
train**': {'**recall**': 89.99}, '**test**': {'**recall**': 87.08}**

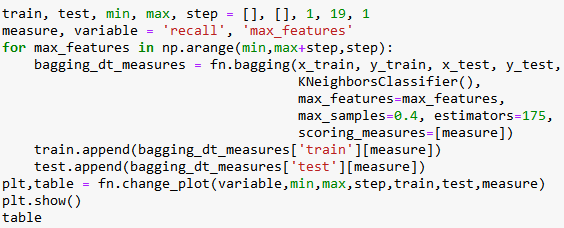
با توجه به اینکه نتیجه تنظیم هایپرپارامترها در این قسمت اختلاف کمی با قسمت قبل دارد ولی مدل کمتر دچار overfitting شده است لذا مبنای ادامه بررسی مقادیر این قسمت قرار داده شد که به شرح زیر می باشد:

**'measures'**: {**'recall'**: 87.07}  
**'params'**:  
 **'n\_estimators'**: 175  
 **'max\_samples'**: 0.4  
 **'max\_features'**: 9

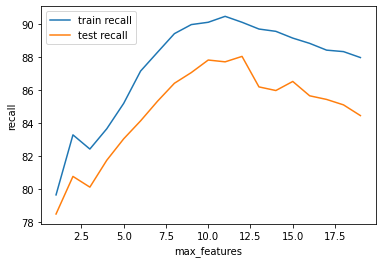
### بررسی تغییر نتایج با تغییر هایپرپارامترها (رفیعی فرد)

با استفاده از کد زیر نمودار تغییرات Recall برای داده آموزشی و تست بر حسب max\_features رسم شد:





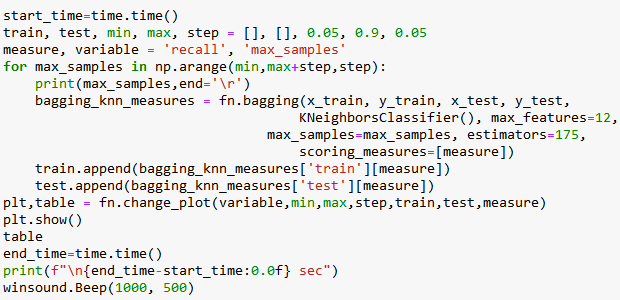
خروجی کد بالا به صورت نمودار و داده جدولی به صورت زیر است:



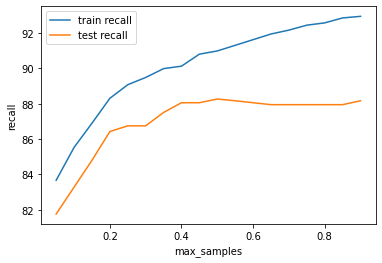
|  |  |  |  |
| --- | --- | --- | --- |
| max\_features | Train | Test | Difference |
| 1 | 79.66 | 78.50 | 1.16 |
| 2 | 83.30 | 80.78 | 2.52 |
| 3 | 82.44 | 80.13 | 2.31 |
| 4 | 83.67 | 81.76 | 1.91 |
| 5 | 85.21 | 83.06 | 2.15 |
| 6 | 87.17 | 84.15 | 3.02 |
| 7 | 88.31 | 85.34 | 2.97 |
| 8 | 89.44 | 86.43 | 3.01 |
| 9 | 89.99 | 87.08 | 2.91 |
| 10 | 90.13 | 87.84 | 2.29 |
| 11 | 90.49 | 87.73 | 2.76 |
| 12 | 90.13 | 88.06 | 2.07 |
| 13 | 89.72 | 86.21 | 3.51 |
| 14 | 89.58 | 85.99 | 3.59 |
| 15 | 89.17 | 86.54 | 2.63 |
| 16 | 88.85 | 85.67 | 3.18 |
| 17 | 88.44 | 85.45 | 2.99 |
| 18 | 88.35 | 85.12 | 3.23 |
| 19 | 87.99 | 84.47 | 3.52 |

ملاحظه می شود که خروجی مدل برای داده تست تا max\_feature=12 افزایش داشته و از آنجا به بعد رو به کاهش می گذارد و ضمن اینکه مقدار overfitting مدل نسبت به قبل 0.9 کاهش داشته معیار Recall هم بهبود داشته است. بنابراین مقدار 12 به عنوان مقدار نهایی برای هایپرپارامتر max\_features در نظر گرفته شد.

پس از آن نمودار تغییرات معیار Recall بر حسب max\_samples رسم شد که کد آن به صورت زیر می باشد:



نتیجه این کد در نمودار و جدول زیر آمده است:

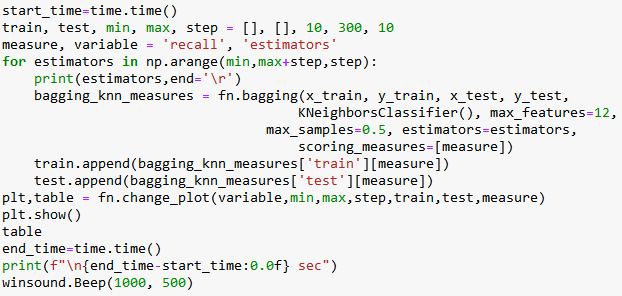


|  |  |  |  |
| --- | --- | --- | --- |
| max\_samples | Train | Test | Difference |
| 0.05 | 83.67 | 81.76 | 1.91 |
| 0.10 | 85.53 | 83.28 | 2.25 |
| 0.15 | 86.90 | 84.80 | 2.10 |
| 0.20 | 88.31 | 86.43 | 1.88 |
| 0.30 | 89.49 | 86.75 | 2.74 |
| 0.35 | 89.99 | 87.51 | 2.48 |
| 0.40 | 90.13 | 88.06 | 2.07 |
| 0.45 | 90.81 | 88.06 | 2.75 |
| 0.50 | 90.99 | 88.27 | 2.72 |
| 0.55 | 91.31 | 88.17 | 3.14 |
| 0.60 | 91.63 | 88.06 | 3.57 |
| 0.70 | 92.17 | 87.95 | 4.22 |
| 0.75 | 92.45 | 87.95 | 4.50 |
| 0.85 | 92.86 | 87.95 | 4.91 |
| 0.90 | 92.95 | 88.17 | 4.78 |

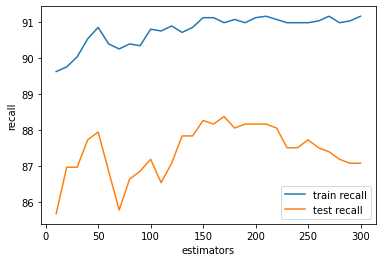
همانطور که از نمودار مشخص است با max\_samples=0.5 مقدار Recall به اندازه حدود 0.2 بهبود پیدا کرده است.

بنابراین مقدار نهایی پارامتر max\_samples برابر با 0.5 در نظر گرفته شد. البته گرچه مدل کمی overfit شده است اما اولاً بالاتر بودن مقدار Recall مهمتر از overfit شدن است و ثانیاً مدل کمی بیشتر از حالت قبل overfit شده است.

در ادامه هایپرپارامتر estimators بررسی شد. از کد زیر جهت رسم نمودار Recall بر حسب تعداد estimator ها استفاده شد:



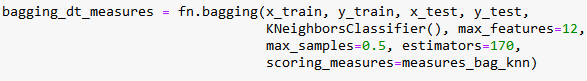
نتایج آن به شرح زیر می باشد:



|  |  |  |  |
| --- | --- | --- | --- |
| estimators | Train | Test | Difference |
| 10 | 89.63 | 85.67 | 3.96 |
| 20 | 89.76 | 86.97 | 2.79 |
| 30 | 90.04 | 86.97 | 3.07 |
| 40 | 90.54 | 87.73 | 2.81 |
| 50 | 90.86 | 87.95 | 2.91 |
| 60 | 90.40 | 86.86 | 3.54 |
| 70 | 90.26 | 85.78 | 4.48 |
| 80 | 90.40 | 86.64 | 3.76 |
| 90 | 90.35 | 86.86 | 3.49 |
| 100 | 90.81 | 87.19 | 3.62 |
| 110 | 90.76 | 86.54 | 4.22 |
| 120 | 90.90 | 87.08 | 3.82 |
| 130 | 90.72 | 87.84 | 2.88 |
| 140 | 90.86 | 87.84 | 3.02 |
| 150 | 91.13 | 88.27 | 2.86 |
| 160 | 91.13 | 88.17 | 2.96 |
| 170 | 90.99 | 88.38 | 2.61 |
| 180 | 91.08 | 88.06 | 3.02 |
| 190 | 90.99 | 88.17 | 2.82 |
| 200 | 91.13 | 88.17 | 2.96 |
| 210 | 91.17 | 88.17 | 3.00 |
| 220 | 91.08 | 88.06 | 3.02 |
| 230 | 90.99 | 87.51 | 3.48 |
| 240 | 90.99 | 87.51 | 3.48 |
| 250 | 90.99 | 87.73 | 3.26 |
| 260 | 91.04 | 87.51 | 3.53 |
| 270 | 91.17 | 87.40 | 3.77 |
| 280 | 90.99 | 87.19 | 3.80 |
| 290 | 91.04 | 87.08 | 3.96 |
| 300 | 91.17 | 87.08 | 4.09 |

با بررسی تغییرات تعداد estimator ها مشاهده شد که در تعداد 170 هم نتیجه حدود 0.1 بهبود یافت و هم overfitting اندکی کاهش یافت. بنابراین مقدار نهایی هایپرپارامتر estimator برابر با 170 شد.

از جمع بندی مراحل بالا مقادیر نهایی و بهینه هایپرپارامترها برای این دسته بند به دست آمد که توسط کد زیر مقدار Recall در حالت بهینه محاسبه شد:



که نتایج به شرح زیر هستند:

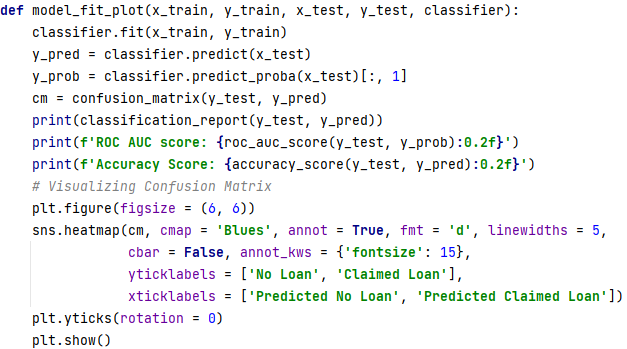
**'train'**: {**'recall'**: 90.99}, **'test'**: {**'recall'**: 88.38}

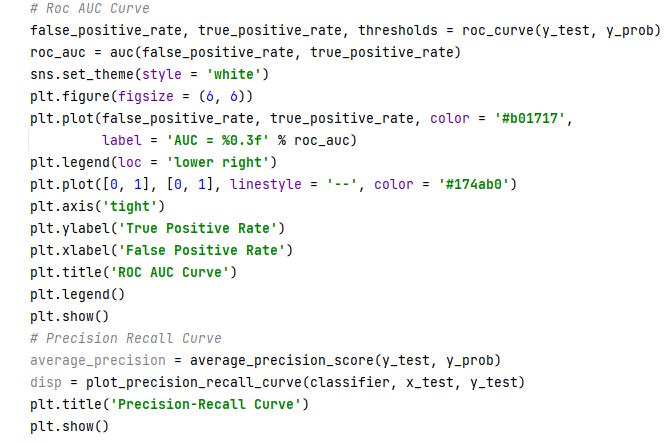
### بهترین هایپرپارامترها و رسم نمودار نتایج بهینه (رفیعی فرد)

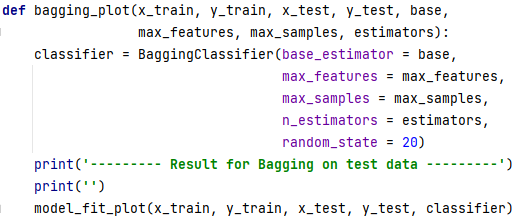
با توجه به نتایج حاصل از جستجوی تصادفی در فضای هایپرپارامترها و نمودارهای بالا بهترین هایپرپارامترها به شرح زیر هستند:

**max\_features**: 12  
**max\_samples**: 0.5  
**n\_estimators**: 170

جهت رسم ماتریس آشفتگی، نمودار ROC AUC و نمودار Precision بر حسب Recall از کد زیر استفاده شد:



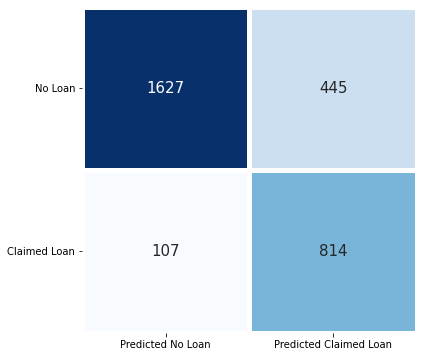
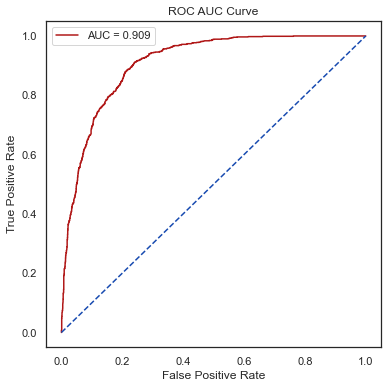


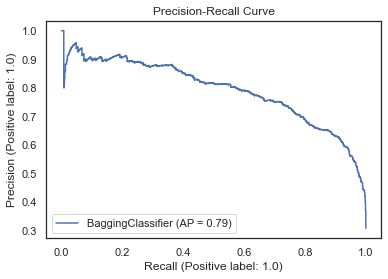




نتایج به صورت زیر هستند:

--------- Result **for** Bagging on test data ---------  
  
 precision recall f1-score support  
  
 0.0 0.94 0.79 0.85 2072  
 1.0 **0.65 0.88 0.75** 921  
  
 accuracy **0.82** 2993  
 macro avg 0.79 0.83 0.80 2993  
weighted avg 0.85 0.82 0.82 2993  
  
ROC AUC score: **0.91**  
Accuracy Score: 0.82



## **ساخت مدل با جنگل تصادفی و تنظیم هایپرپارامترها** (رفیعی فرد)

متد ساخت مدل:

classifier = RandomForestClassifier (criterion, max\_depth, max\_features,

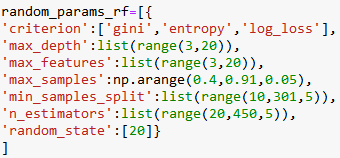
max\_samples, min\_samples\_split, n\_estimators, random\_state=20)

### معرفی هایپرپارمترها (رفیعی فرد)

1. criterion معیار اندازه گیری خطا
2. max\_depth حداکثر عمق
3. max\_features حداکثر تعداد ویژگی هایی که در هر بار ساخت مدل با دسته بند پایه (در اینجا درخت تصمیم) استفاده می شود
4. max\_samples حداکثر تعداد نمونه هایی که در در هر بار ساخت مدل با دسته بند پایه (در اینجا درخت تصمیم) استفاده می شود
5. min\_samples\_split حداقل تعداد داده در هر گره
6. n\_estimators تعداد دفعات ساخت مدل درخت تصمیم

### تنظیم پارامترها برای به دست آوردن بهترین حالت بر حسب معیار Recall (رفیعی فرد)

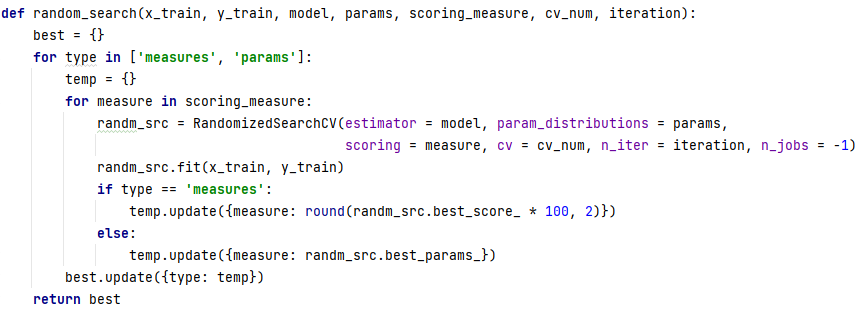
در ادامه بازه تغییر هایپرپارامترها برای استفاده از متد RandomizedSearchCV مشخص شد:



جهت طولانی نشدن زمان اجرای کدها فقط معیار Recall محاسبه گردید که با کد زیر معیار مورد نظر را مشخص می کنیم:

measures\_rf = [**'recall'**]

ابتدا تابع زیر جهت یافتن حالت بهینه بر اساس معیار مورد نظر استفاده شد:



سپس با استفاده از کد زیر با 5 بار اجرای تابع فوق بهترین حالت بر حسب معیار recall به دست آمد:



نتایج به شرح زیر می باشد:

**try** 0 Recall:85.3 criterion: log\_loss max\_depth:17 min\_samples\_split:15  
 max\_features: 16 max\_samples: 0.65 n\_estimators: 325  
**try** 1  
**try** 2 Recall:85.44 criterion: entropy max\_depth:17 min\_samples\_split:10  
 max\_features: 16 max\_samples: 0.40 n\_estimators: 50  
**try** 3  
**try** 4 Recall:86.26 criterion: log\_loss max\_depth:12 min\_samples\_split:25  
 max\_features: 4 max\_samples: 0.50 n\_estimators: 285  
**try** 5  
**try** 6  
**try** 7  
**try** 8  
**try** 9  
1676 sec

**'measures'**: {**'recall'**: 86.26}  
**'params'**:

**criterion**: log\_loss

**max\_depth:** 12

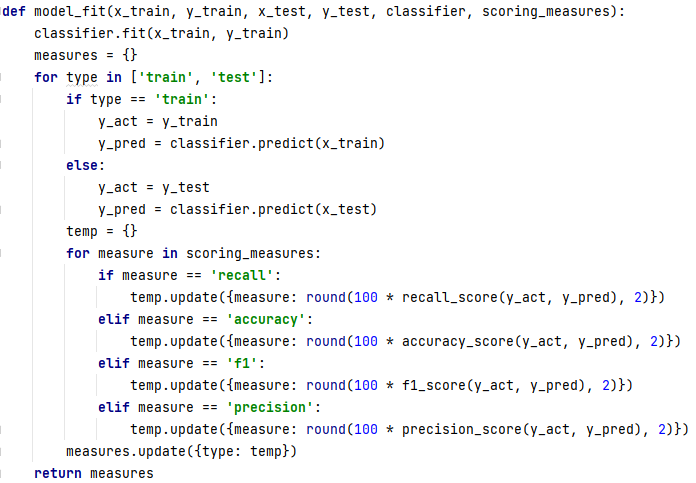
**min\_samples\_split:** 25

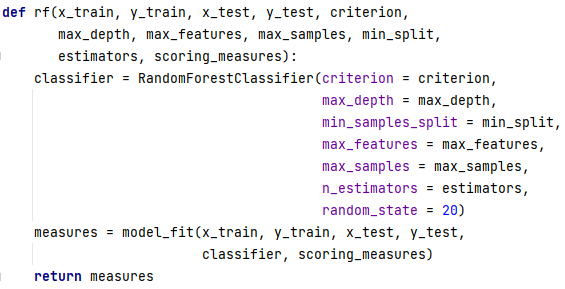
**max\_features**: 4

**max\_samples**: 0.5  
 **n\_estimators**: 285

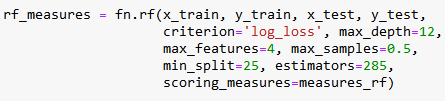
### ارزیابی مدل بر روی داده آموزشی و تست (رفیعی فرد)

دو تابع زیر برای ارزیابی مدل استفاده شد:



****

سپس با کد زیر مقدار معیار recall برای داده آموزشی و تست محاسبه شد:



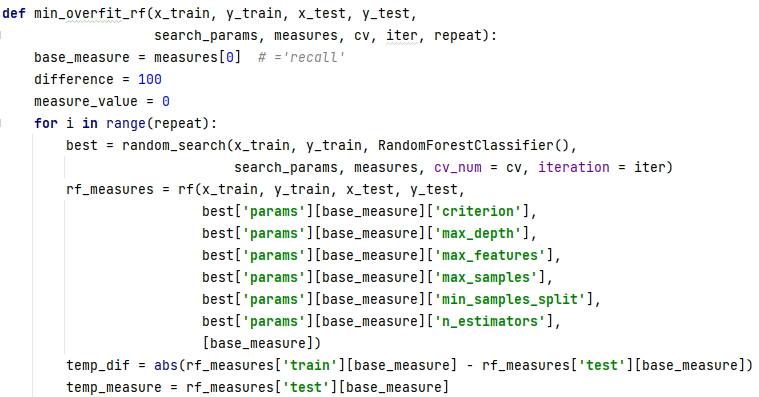
نتایج به صورت زیر به دست آمد:

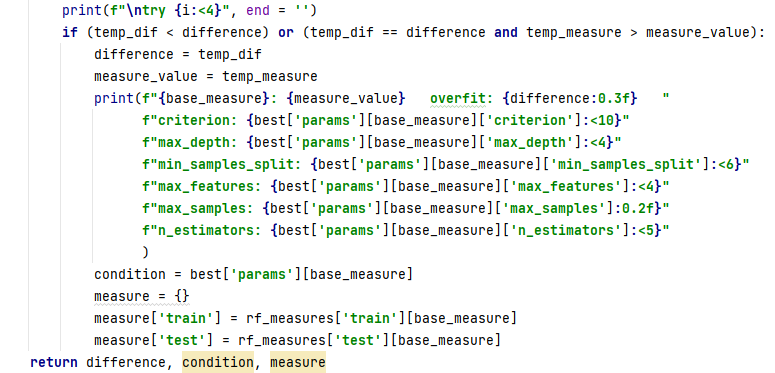
**'train'**: {**'recall'**: 88.03}, **'test'**: {**'recall'**: 86.1}

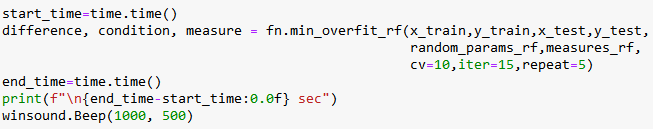
بر اساس این نتایج مدل تا حدی دچار overfitting شده است که در ادامه سعی شده است نتایج بهتری از این حیث حاصل شود.

### جستجوی کمترین حالت overfitting (رفیعی فرد)

تابع زیر برای یافتن کمترین اختلاف معیار Recall بین داده آموزشی و تست استفاده شد:



 سپس با استفاده از کد زیر، حالتی را که کمترین overfitting اتفاق بیفتد به دست آمد:

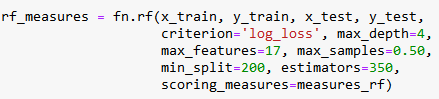


نتایج به شرح زیر است:

**try** 0 recall: 85.67 overfite: 2.680 criterion: gini max\_depth: 15 min\_samples\_split: 20  
 max\_features: 6 max\_samples: 0.40 n\_estimators: 150   
  
**try** 1   
**try** 2   
**try** 3 recall: 85.88 overfite: 0.440 criterion: entropy max\_depth: 19 min\_samples\_split: 210  
 max\_features: 19 max\_samples: 0.50 n\_estimators: 300   
  
**try** 4 recall: 84.04 overfite: 0.350 criterion: log\_loss max\_depth: 4 min\_samples\_split: 200  
 max\_features: 17 max\_samples: 0.50 n\_estimators: 350   
  
1359 sec

در هر بار سعی، اگر نتایج بهتر شده باشد آن را چاپ می کند. مقادیر Recall روی داده تست هستند. ملاحظه می شود که در بهترین حالت دارای overfitting برابر با 0.35 است که مقدار قابل قبولی است.

با استفاده از دستور زیر نتایج برای داده آموزشی و تست بر اساس معیار Recall نمایش داده می شود:

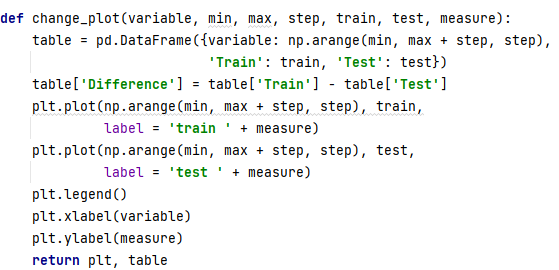


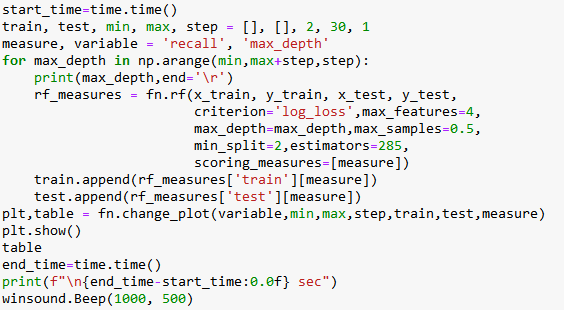
نتایج به شرح زیر است:

train**': {'**recall**': 84.39}, '**test**': {'**recall**': 84.04}**

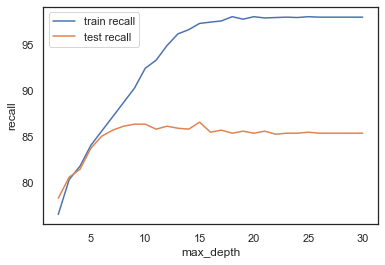
### بررسی تغییر نتایج با تغییر هایپرپارامترها (رفیعی فرد)

با استفاده از کد زیر نمودار تغییرات Recall برای داده آموزشی و تست بر حسب max\_depth رسم شد:





خروجی کد بالا به صورت نمودار و داده جدولی به صورت زیر است:



|  |  |  |  |
| --- | --- | --- | --- |
| max\_depth | Train | Test | Difference |
| 2 | 76.52 | 78.28 | -1.76 |
| 3 | 80.30 | 80.56 | -0.26 |
| 4 | 81.76 | 81.43 | 0.33 |
| 5 | 84.03 | 83.71 | 0.32 |
| 6 | 85.58 | 85.02 | 0.56 |
| 7 | 87.12 | 85.67 | 1.45 |
| 8 | 88.67 | 86.10 | 2.57 |
| 9 | 90.22 | 86.32 | 3.90 |
| 10 | 92.40 | 86.32 | 6.08 |
| 11 | 93.27 | 85.78 | 7.49 |
| 12 | 94.86 | 86.10 | 8.76 |
| 13 | 96.13 | 85.88 | 10.25 |
| 14 | 96.59 | 85.78 | 10.81 |
| 15 | 97.27 | 86.54 | 10.73 |
| 16 | 97.41 | 85.45 | 11.96 |
| 17 | 97.54 | 85.67 | 11.87 |
| 18 | 98.00 | 85.34 | 12.66 |
| 19 | 97.73 | 85.56 | 12.17 |
| 20 | 98.00 | 85.34 | 12.66 |
| 21 | 97.86 | 85.56 | 12.30 |
| 22 | 97.91 | 85.23 | 12.68 |
| 23 | 97.95 | 85.34 | 12.61 |
| 24 | 97.91 | 85.34 | 12.57 |
| 25 | 98.00 | 85.45 | 12.55 |
| 26 | 97.95 | 85.34 | 12.61 |
| 27 | 97.95 | 85.34 | 12.61 |
| 28 | 97.95 | 85.34 | 12.61 |
| 29 | 97.95 | 85.34 | 12.61 |
| 30 | 97.95 | 85.34 | 12.61 |

بهترین نتیجه برای Recall برابر است با 86.54 که کمی نسبت به مرحله قبل بهبود یافته است. بنابراین مقدار 15 به عنوان مقدار نهایی برای هایپرپارامتر max\_depth در نظر گرفته شد.

### بهترین هایپرپارامترها و رسم نمودار نتایج بهینه (رفیعی فرد)

با توجه به نتایج حاصل از جستجوی تصادفی در فضای هایپرپارامترها و نمودارهای بالا بهترین هایپرپارامترها به شرح زیر هستند:

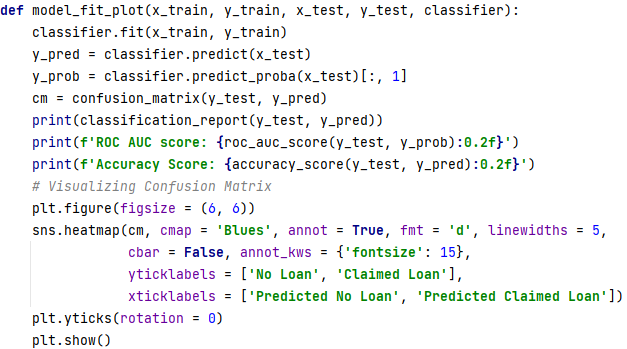
**criterion**: log\_loss

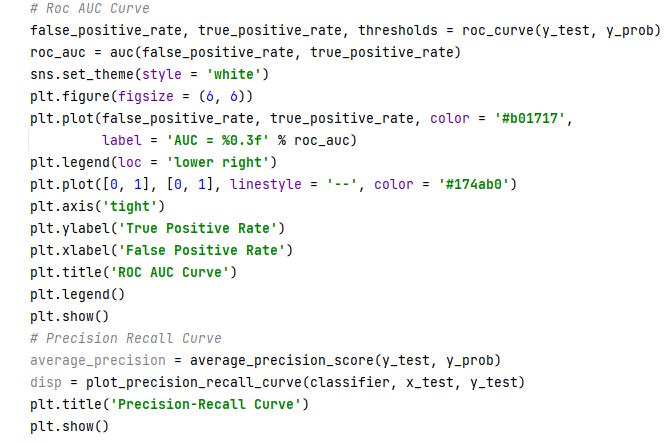
**max\_depth**: 15

**max\_features**: 4  
**max\_samples**: 0.5

**min\_split**: 2  
**n\_estimators**: 285

جهت رسم ماتریس آشفتگی، نمودار ROC AUC و نمودار Precision بر حسب Recall از کد زیر استفاده شد:



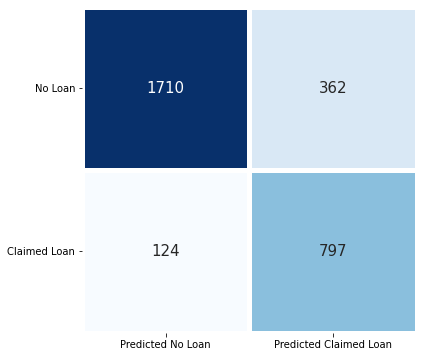
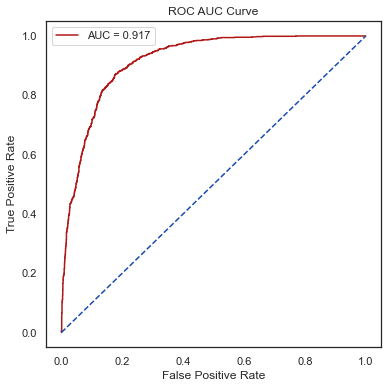


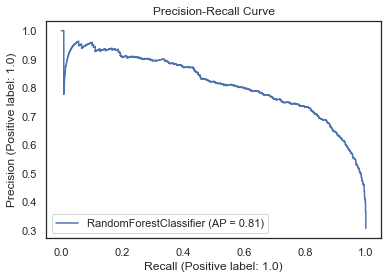
**def** rf\_plot(x\_train, y\_train, x\_test, y\_test,  
 criterion, max\_depth, max\_features,  
 max\_samples, min\_split, estimators):  
 classifier = RandomForestClassifier(criterion = criterion,  
 max\_depth = max\_depth,  
 min\_samples\_split = min\_split,  
 max\_features = max\_features,  
 max\_samples = max\_samples,  
 n\_estimators = estimators,  
 random\_state = 20)  
 print(**'--------- Result for Random Forest on test data ---------'**)  
 print(**''**)  
 model\_fit\_plot(x\_train, y\_train, x\_test, y\_test, classifier)

fn.rf\_plot(x\_train, y\_train, x\_test, y\_test,  
 criterion=**'log\_loss'**, max\_depth=4,  
 max\_features=17,max\_samples=0.5,  
 min\_split=200, estimators=350)

نتایج به صورت زیر هستند:

--------- Result **for** Random Forest on test data ---------  
  
 precision recall f1-score support  
  
 0.0 0.93 0.83 0.88 2072  
 1.0 **0.69 0.87 0.77** 921  
  
 accuracy **0.84** 2993  
 macro avg 0.81 0.85 0.82 2993  
weighted avg 0.86 0.84 0.84 2993  
  
ROC AUC score: **0.92**  
Accuracy Score: 0.84



## **ساخت مدل با آدابوست و تنظیم هایپرپارامترها** (رفیعی فرد)

متد ساخت مدل:

classifier = AdaBoostClassifier (algorithm, learning\_rate, n\_estimators, random\_state=20)

### معرفی هایپرپارمترها (رفیعی فرد)

1. algorithm الگوریتم مورد استفاده برای دسته بند که می تواند ‘SAMME’ یا ‘SAMME.R’ باشد
2. learning\_rate وزن اعمال شده به هر دسته بند در هر بار ساخت مدل
3. n\_estimators تعداد دفعات ساخت مدل که به صورت پیش فرض درخت تصمیم است.

### تنظیم پارامترها برای به دست آوردن بهترین حالت بر حسب معیار Recall (رفیعی فرد)

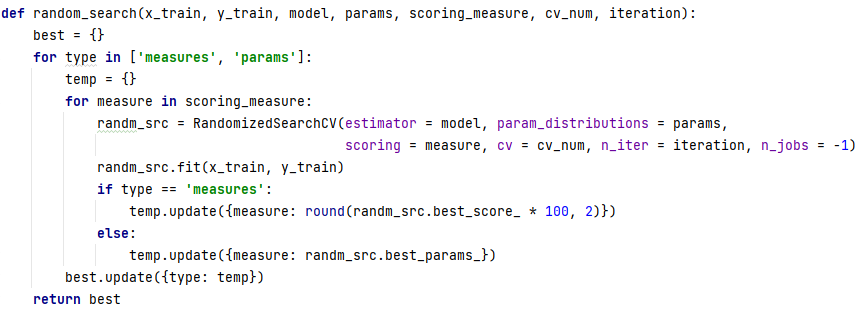
در ادامه بازه تغییر هایپرپارامترها برای استفاده از متد RandomizedSearchCV مشخص شد:

random\_params\_ab=[{  
**'algorithm'**:[**'SAMME'**, **'SAMME.R'**],  
**'learning\_rate'**:list(np.arange(0.05,0.99,0.2)),  
**'n\_estimators'**:list(np.arange(20,410,10)),  
**'random\_state'**:[20]}]

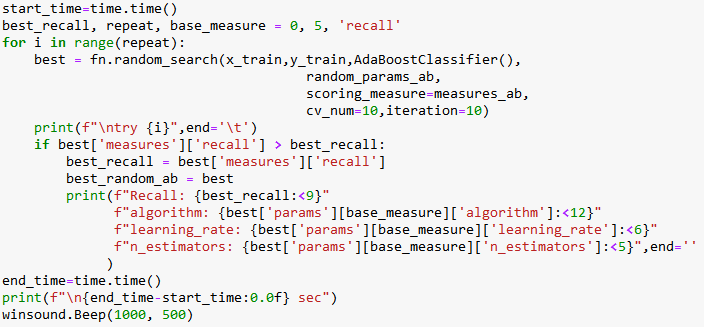
جهت طولانی نشدن زمان اجرای کدها فقط معیار Recall محاسبه گردید که با کد زیر معیار مورد نظر را مشخص می کنیم:

measures\_rf = [**'recall'**]

ابتدا تابع زیر جهت یافتن حالت بهینه بر اساس معیار مورد نظر استفاده شد:



سپس با استفاده از کد زیر با 5 بار اجرای تابع فوق بهترین حالت بر حسب معیار recall به دست آمد:



نتایج به شرح زیر می باشد:

**try** 0 Recall:85.94 algorithm:SAMME.R learning\_rate:0.05 n\_estimators: 270   
**try** 1 Recall:86.62 algorithm:SAMME learning\_rate:0.25 n\_estimators: 310   
**try** 2   
**try** 3   
**try** 4   
547 sec

**'measures'**: {**'recall'**: 86.62}  
**'params'**:

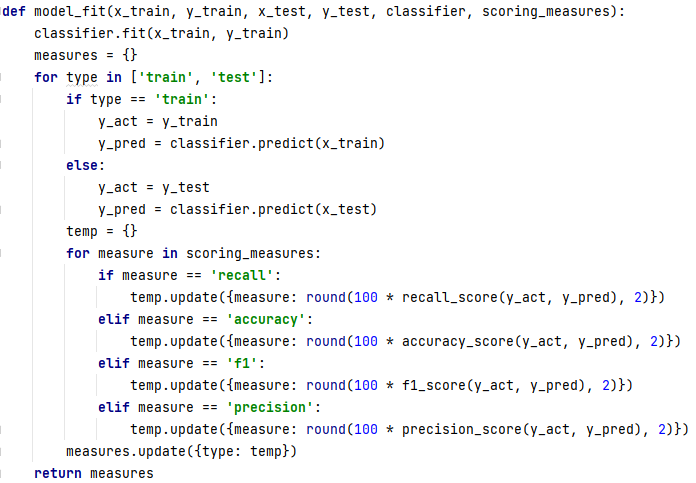
**algorithm**: SAMME

**learning\_rate:** 0.25

**n\_estimators:** 310

### ارزیابی مدل بر روی داده آموزشی و تست (رفیعی فرد)

دو تابع زیر برای ارزیابی مدل استفاده شد:



**def** ab(x\_train, y\_train, x\_test, y\_test, algorithm,  
 learning\_rate, estimators, scoring\_measures):  
 classifier = AdaBoostClassifier(algorithm = algorithm,  
 learning\_rate = learning\_rate,  
 n\_estimators = estimators,  
 random\_state = 20)  
 measures = model\_fit(x\_train, y\_train, x\_test, y\_test,  
 classifier, scoring\_measures)  
 **return** measures

سپس با کد زیر مقدار معیار recall برای داده آموزشی و تست محاسبه شد:

ab\_measure = fn.ab(x\_train,y\_train,x\_test,y\_test,  
 algorithm=**'SAMME'**,estimators=310,  
 learning\_rate=0.25,  
 scoring\_measures=measures\_ab)

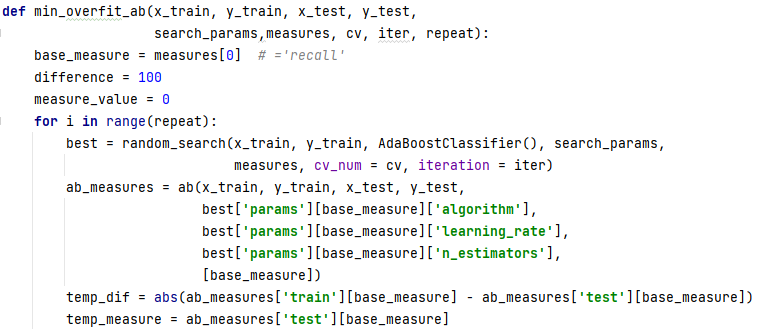
نتایج به صورت زیر به دست آمد:

**'train'**: {**'recall'**: 86.21}, **'test'**: {**'recall'**: 85.88}

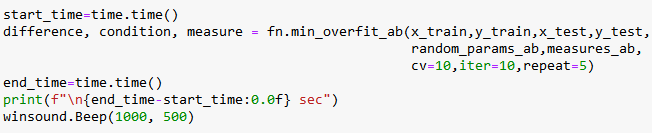
بر اساس این نتایج مدل کمی دچار overfitting شده است که در ادامه سعی شده است نتایج بهتری از این حیث حاصل شود.

### جستجوی کمترین حالت overfitting (رفیعی فرد)

تابع زیر برای یافتن کمترین اختلاف معیار Recall بین داده آموزشی و تست استفاده شد:



سپس با استفاده از کد زیر، حالتی را که کمترین overfitting اتفاق بیفتد به دست آمد:



نتایج به شرح زیر است:

**try** 0 recall: 85.88 overfit: 0.470 algorithm: SAMME.R

learning\_rate: 0.25 n\_estimators: 50   
**try** 1   
**try** 2 recall: 86.21 overfit: 0.050 algorithm: SAMME.R

learning\_rate: 0.25 n\_estimators: 30   
**try** 3   
**try** 4   
510 sec

در هر بار سعی، اگر نتایج بهتر شده باشد آن را چاپ می کند. مقادیر Recall روی داده تست هستند. ملاحظه می شود که مقدار overfitting بسیار کاهش یافته و به عدد 0.05 رسیده

با استفاده از دستور زیر نتایج برای داده آموزشی و تست بر اساس معیار Recall نمایش داده می شود:

ab\_measure = fn.ab(x\_train,y\_train,x\_test,y\_test,  
 algorithm=**'SAMME.R'**,estimators=30,  
 learning\_rate=0.25,  
 scoring\_measures=measures\_ab)

نتایج به شرح زیر است:

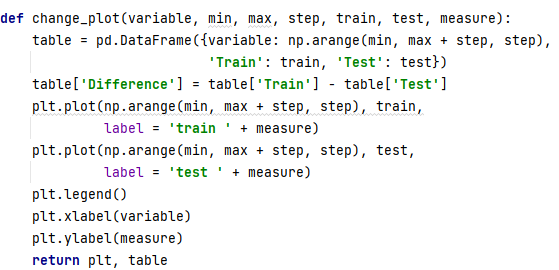
train**': {'**recall**': 86.26}, '**test**': {'**recall**': 86.21}**

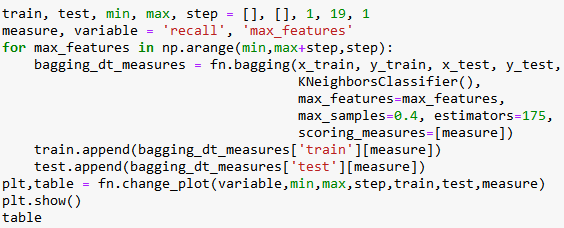
با توجه به اینکه نتیجه تنظیم هایپرپارامترها در این قسمت اختلاف کمی با قسمت قبل دارد ولی مدل کمتر دچار overfitting شده است لذا مبنای ادامه بررسی مقادیر این قسمت قرار داده شد که به شرح زیر می باشد:

**'measures'**: {**'recall'**: 87.07}  
**'params'**:  
 **'n\_estimators'**: 175  
 **'max\_samples'**: 0.4  
 **'max\_features'**: 9

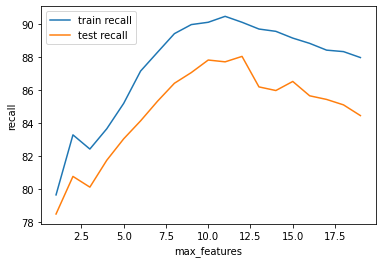
### بررسی تغییر نتایج با تغییر هایپرپارامترها (رفیعی فرد)

با استفاده از کد زیر نمودار تغییرات Recall برای داده آموزشی و تست بر حسب max\_features رسم شد:





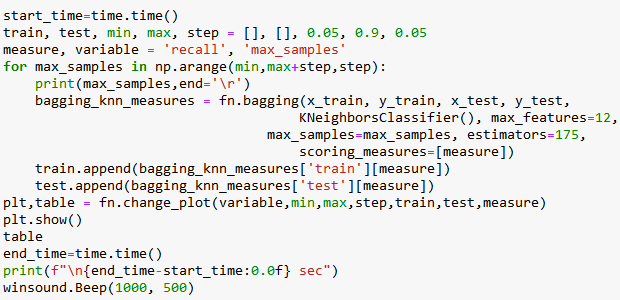
خروجی کد بالا به صورت نمودار و داده جدولی به صورت زیر است:



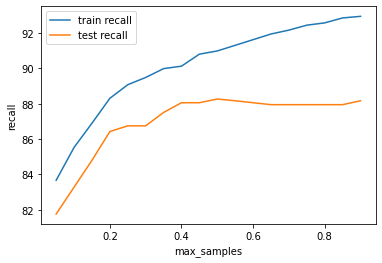
|  |  |  |  |
| --- | --- | --- | --- |
| max\_features | Train | Test | Difference |
| 1 | 79.66 | 78.50 | 1.16 |
| 2 | 83.30 | 80.78 | 2.52 |
| 3 | 82.44 | 80.13 | 2.31 |
| 4 | 83.67 | 81.76 | 1.91 |
| 5 | 85.21 | 83.06 | 2.15 |
| 6 | 87.17 | 84.15 | 3.02 |
| 7 | 88.31 | 85.34 | 2.97 |
| 8 | 89.44 | 86.43 | 3.01 |
| 9 | 89.99 | 87.08 | 2.91 |
| 10 | 90.13 | 87.84 | 2.29 |
| 11 | 90.49 | 87.73 | 2.76 |
| 12 | 90.13 | 88.06 | 2.07 |
| 13 | 89.72 | 86.21 | 3.51 |
| 14 | 89.58 | 85.99 | 3.59 |
| 15 | 89.17 | 86.54 | 2.63 |
| 16 | 88.85 | 85.67 | 3.18 |
| 17 | 88.44 | 85.45 | 2.99 |
| 18 | 88.35 | 85.12 | 3.23 |
| 19 | 87.99 | 84.47 | 3.52 |

ملاحظه می شود که خروجی مدل برای داده تست تا max\_feature=12 افزایش داشته و از آنجا به بعد رو به کاهش می گذارد و ضمن اینکه مقدار overfitting مدل نسبت به قبل 0.9 کاهش داشته معیار Recall هم بهبود داشته است. بنابراین مقدار 12 به عنوان مقدار نهایی برای هایپرپارامتر max\_features در نظر گرفته شد.

پس از آن نمودار تغییرات معیار Recall بر حسب max\_samples رسم شد که کد آن به صورت زیر می باشد:



نتیجه این کد در نمودار و جدول زیر آمده است:

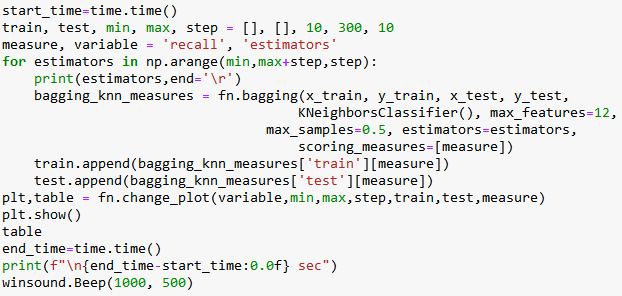


|  |  |  |  |
| --- | --- | --- | --- |
| max\_samples | Train | Test | Difference |
| 0.05 | 83.67 | 81.76 | 1.91 |
| 0.10 | 85.53 | 83.28 | 2.25 |
| 0.15 | 86.90 | 84.80 | 2.10 |
| 0.20 | 88.31 | 86.43 | 1.88 |
| 0.30 | 89.49 | 86.75 | 2.74 |
| 0.35 | 89.99 | 87.51 | 2.48 |
| 0.40 | 90.13 | 88.06 | 2.07 |
| 0.45 | 90.81 | 88.06 | 2.75 |
| 0.50 | 90.99 | 88.27 | 2.72 |
| 0.55 | 91.31 | 88.17 | 3.14 |
| 0.60 | 91.63 | 88.06 | 3.57 |
| 0.70 | 92.17 | 87.95 | 4.22 |
| 0.75 | 92.45 | 87.95 | 4.50 |
| 0.85 | 92.86 | 87.95 | 4.91 |
| 0.90 | 92.95 | 88.17 | 4.78 |

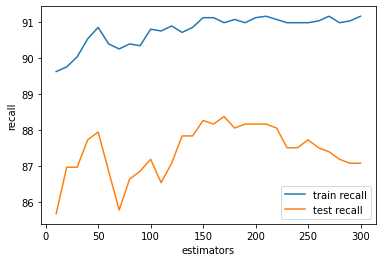
همانطور که از نمودار مشخص است با max\_samples=0.5 مقدار Recall به اندازه حدود 0.2 بهبود پیدا کرده است.

بنابراین مقدار نهایی پارامتر max\_samples برابر با 0.5 در نظر گرفته شد. البته گرچه مدل کمی overfit شده است اما اولاً بالاتر بودن مقدار Recall مهمتر از overfit شدن است و ثانیاً مدل کمی بیشتر از حالت قبل overfit شده است.

در ادامه هایپرپارامتر estimators بررسی شد. از کد زیر جهت رسم نمودار Recall بر حسب تعداد estimator ها استفاده شد:



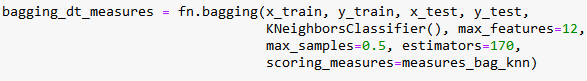
نتایج آن به شرح زیر می باشد:



|  |  |  |  |
| --- | --- | --- | --- |
| estimators | Train | Test | Difference |
| 10 | 89.63 | 85.67 | 3.96 |
| 20 | 89.76 | 86.97 | 2.79 |
| 30 | 90.04 | 86.97 | 3.07 |
| 40 | 90.54 | 87.73 | 2.81 |
| 50 | 90.86 | 87.95 | 2.91 |
| 60 | 90.40 | 86.86 | 3.54 |
| 70 | 90.26 | 85.78 | 4.48 |
| 80 | 90.40 | 86.64 | 3.76 |
| 90 | 90.35 | 86.86 | 3.49 |
| 100 | 90.81 | 87.19 | 3.62 |
| 110 | 90.76 | 86.54 | 4.22 |
| 120 | 90.90 | 87.08 | 3.82 |
| 130 | 90.72 | 87.84 | 2.88 |
| 140 | 90.86 | 87.84 | 3.02 |
| 150 | 91.13 | 88.27 | 2.86 |
| 160 | 91.13 | 88.17 | 2.96 |
| 170 | 90.99 | 88.38 | 2.61 |
| 180 | 91.08 | 88.06 | 3.02 |
| 190 | 90.99 | 88.17 | 2.82 |
| 200 | 91.13 | 88.17 | 2.96 |
| 210 | 91.17 | 88.17 | 3.00 |
| 220 | 91.08 | 88.06 | 3.02 |
| 230 | 90.99 | 87.51 | 3.48 |
| 240 | 90.99 | 87.51 | 3.48 |
| 250 | 90.99 | 87.73 | 3.26 |
| 260 | 91.04 | 87.51 | 3.53 |
| 270 | 91.17 | 87.40 | 3.77 |
| 280 | 90.99 | 87.19 | 3.80 |
| 290 | 91.04 | 87.08 | 3.96 |
| 300 | 91.17 | 87.08 | 4.09 |

با بررسی تغییرات تعداد estimator ها مشاهده شد که در تعداد 170 هم نتیجه حدود 0.1 بهبود یافت و هم overfitting اندکی کاهش یافت. بنابراین مقدار نهایی هایپرپارامتر estimator برابر با 170 شد.

از جمع بندی مراحل بالا مقادیر نهایی و بهینه هایپرپارامترها برای این دسته بند به دست آمد که توسط کد زیر مقدار Recall در حالت بهینه محاسبه شد:



که نتایج به شرح زیر هستند:

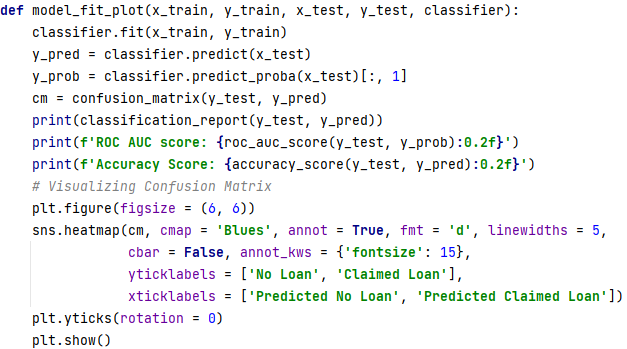
**'train'**: {**'recall'**: 90.99}, **'test'**: {**'recall'**: 88.38}

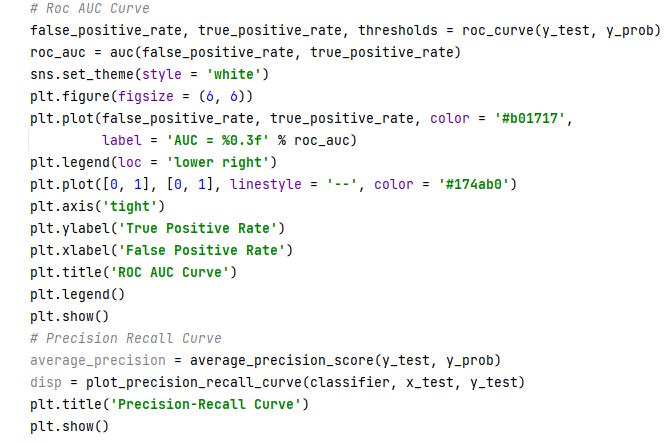
### بهترین هایپرپارامترها و رسم نمودار نتایج بهینه (رفیعی فرد)

با توجه به نتایج حاصل از جستجوی تصادفی در فضای هایپرپارامترها و نمودارهای بالا بهترین هایپرپارامترها به شرح زیر هستند:

**algorithm**: SAMME.R  
**learning\_rate**: 0.25  
**n\_estimators**: 30

جهت رسم ماتریس آشفتگی، نمودار ROC AUC و نمودار Precision بر حسب Recall از کد زیر استفاده شد:



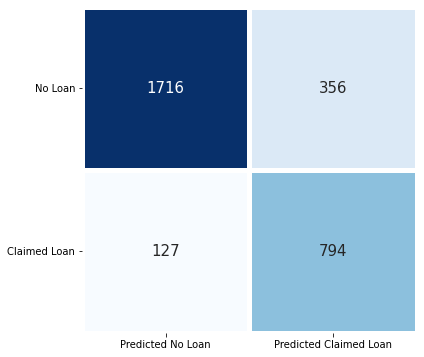
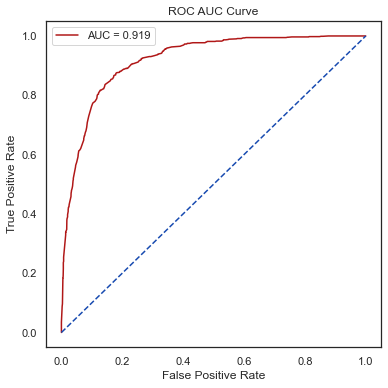


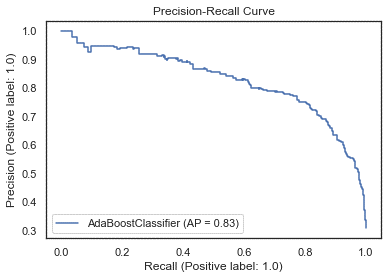
**def** ab\_plot(x\_train, y\_train, x\_test, y\_test,  
 algorithm, learning\_rate, estimators):  
 classifier = AdaBoostClassifier(algorithm = algorithm,  
 learning\_rate = learning\_rate,  
 n\_estimators = estimators,  
 random\_state = 20)  
 print(**'--------- Result for AdaBoost on test data ---------'**)  
 print(**''**)  
 model\_fit\_plot(x\_train, y\_train, x\_test, y\_test, classifier)

fn.ab\_plot(x\_train,y\_train,x\_test,y\_test,  
 algorithm=**'SAMME.R'**,learning\_rate=0.25,  
 estimators=30)

نتایج به صورت زیر هستند:

--------- Result **for** AdaBoost on test data ---------  
  
 precision recall f1-score support  
  
 0.0 0.93 0.83 0.88 2072  
 1.0 **0.69 0.86 0.77** 921  
  
 accuracy **0.84** 2993  
 macro avg 0.81 0.85 0.82 2993  
weighted avg 0.86 0.84 0.84 2993  
  
ROC AUC score: **0.92**  
Accuracy Score: 0.84



## **ساخت مدل با KNN و تنظیم هایپرپارامترها** (رفیعی فرد)

متد ساخت مدل:

**classifier = KNeighborsClassifier(metric, n\_neighbors)**

### معرفی هایپرپارمترها (رفیعی فرد)

1. **metric** تابع فاصله که می تواند 'euclidean' یا 'manhattan' باشد
2. **n\_neighbors** تعداد همسایه ها

### تنظیم پارامترها برای به دست آوردن بهترین حالت بر حسب معیار Recall (رفیعی فرد)

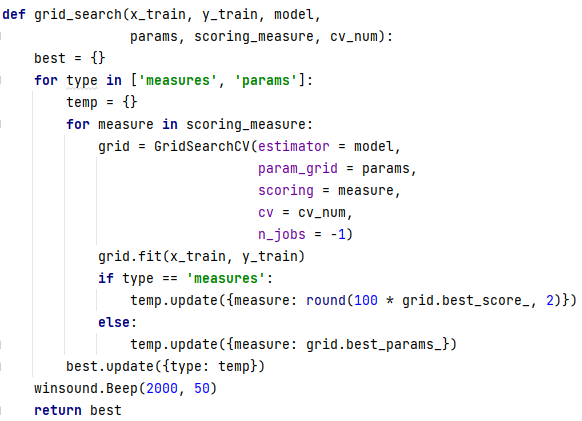
در ادامه بازه تغییر هایپرپارامترها برای استفاده از متد RandomizedSearchCV مشخص شد:

grid\_params\_knn=[{  
**'metric'**: [**'euclidean'**, **'manhattan'**, **'chebyshev'**, **'minkowski'**],  
**'n\_neighbors'**:list(np.arange(3,20,1))  
}]

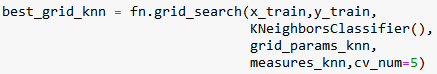
با کد زیر معیارهای مورد نظر را مشخص می کنیم:

measures\_knn = [**'recall'**,**'accuracy'**]

تابع زیر جهت یافتن حالت بهینه بر اساس معیار مورد نظر استفاده شد:



با استفاده از کد زیر بهترین حالت بر حسب معیار recall به دست آمد:



نتایج به شرح زیر می باشد:

**'measures'**: {**'recall'**: 86.62, **'accuracy'**: 81.28}  
**'params'**:

**'recall'**

**metric**: manhattan

**n\_neighbors:** 19

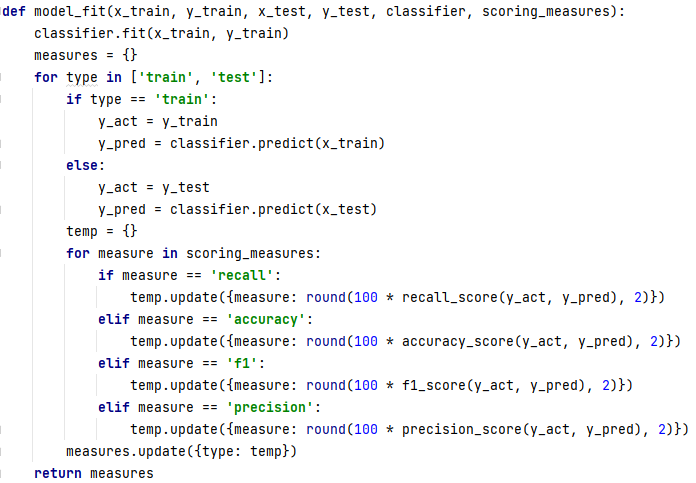
**'accuracy'**

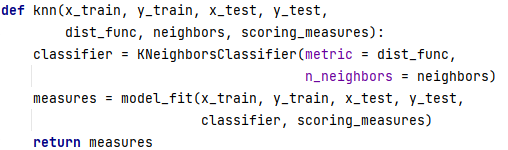
**metric**: manhattan

**n\_neighbors:** 19

### ارزیابی مدل بر روی داده آموزشی و تست (رفیعی فرد)

دو تابع زیر برای ارزیابی مدل استفاده شد:



****

سپس با کد زیر مقدار معیار recall برای داده آموزشی و تست محاسبه شد:



نتایج به صورت زیر به دست آمد:

**'train'**: {**'recall'**: 89.04, **'accuracy'**: 83.23}  
**'test'**: {**'recall'**: 87.3, **'accuracy'**: 79.12}

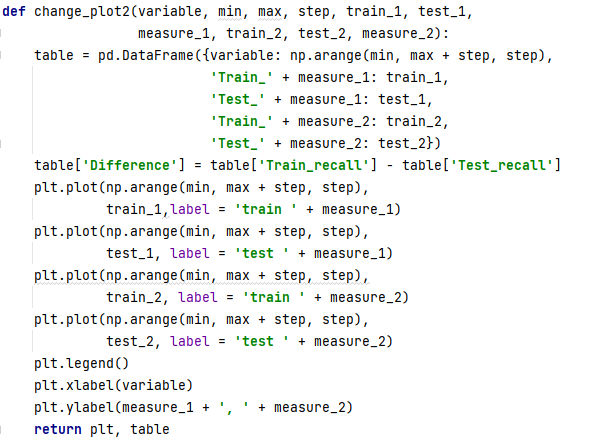
**metric**: manhattan

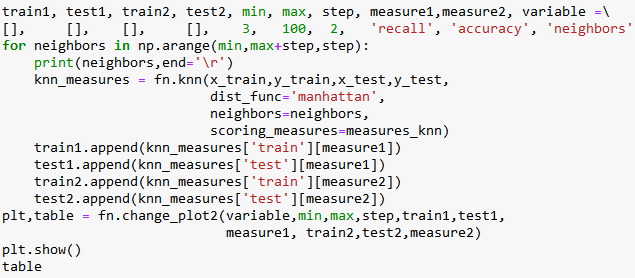
**n\_neighbors:** 19

بر اساس این نتایج مدل کمی دچار overfitting شده است که در ادامه سعی شده است نتایج بهتری از این حیث حاصل شود.

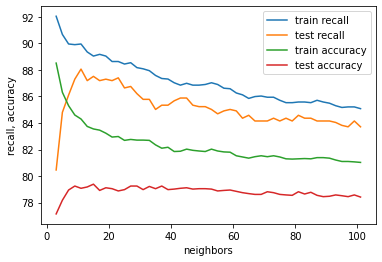
### بررسی تغییر نتایج با تغییر هایپرپارامترها (رفیعی فرد)

با استفاده از کد زیر نمودار تغییرات Recall و Accuracy برای داده آموزشی و تست بر حسب تعداد همسایگان برای تابع فاصله manhattan رسم شد:





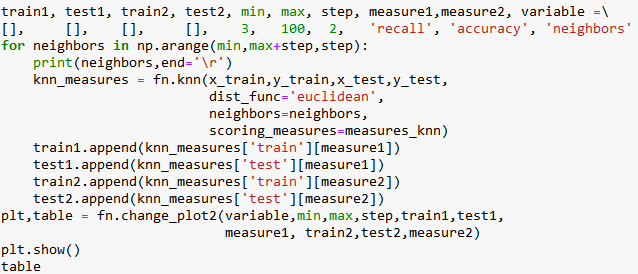
خروجی کد بالا به صورت نمودار و داده جدولی به صورت زیر است:



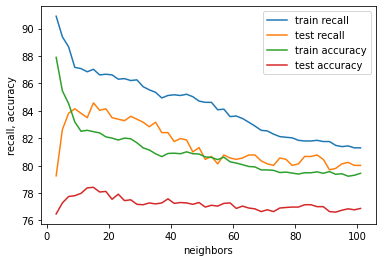
|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| neighbors | Train\_recall | Test\_recall | Train\_accuracy | Test\_accuracy | Difference |
| 3 | 92.04 | 80.46 | 88.51 | 77.15 | 11.58 |
| 5 | 90.67 | 84.80 | 86.31 | 78.18 | 5.87 |
| 7 | 89.95 | 86.10 | 85.30 | 78.95 | 3.85 |
| 9 | 89.90 | 87.30 | 84.60 | 79.25 | 2.60 |
| 11 | 89.95 | 88.06 | 84.30 | 79.08 | 1.89 |
| 13 | 89.35 | 87.19 | 83.74 | 79.18 | 2.16 |
| 15 | 89.04 | 87.51 | 83.55 | 79.39 | 1.53 |
| 17 | 89.17 | 87.19 | 83.46 | 78.92 | 1.98 |
| 25 | 88.44 | 86.64 | 82.69 | 78.98 | 1.80 |
| 27 | 88.54 | 86.75 | 82.76 | 79.25 | 1.79 |
| 39 | 87.31 | 85.34 | 82.17 | 78.98 | 1.97 |
| 47 | 86.85 | 85.34 | 81.94 | 79.02 | 1.51 |
| 51 | 86.90 | 85.23 | 81.85 | 79.05 | 1.67 |
| 63 | 86.12 | 84.36 | 81.44 | 78.75 | 1.76 |
| 75 | 85.71 | 84.15 | 81.44 | 78.62 | 1.56 |
| 89 | 85.58 | 84.15 | 81.39 | 78.45 | 1.43 |
| 97 | 85.21 | 83.71 | 81.10 | 78.45 | 1.50 |
| 101 | 85.08 | 83.71 | 81.03 | 78.42 | 1.37 |

ملاحظه می شود که خروجی مدل برای داده تست تا تعداد همسایگان 11 افزایش داشته و از آنجا به بعد رو به کاهش می گذارد و ضمن اینکه مقدار overfitting مدل در این نقطه نسبت به قبل افزایش ناچیزی داشته است. بنابراین مقدار 11 به عنوان مقدار نهایی برای هایپرپارامتر تعداد همسایگان در نظر گرفته شد.

در ادامه منحنی تغییرات Recall بر حسب تعداد همسایگان برای تابع فاصله euclidean رسم شد.



نتیجه این کد در نمودار و جدول زیر آمده است:



|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| neighbors | Train\_recall | Test\_recall | Train\_accuracy | Test\_accuracy | Difference |
| 3 | 90.90 | 79.26 | 87.90 | 76.48 | 11.64 |
| 5 | 89.40 | 82.63 | 85.46 | 77.28 | 6.77 |
| 7 | 88.67 | 83.82 | 84.53 | 77.75 | 4.85 |
| 9 | 87.17 | 84.15 | 83.19 | 77.81 | 3.02 |
| 11 | 87.08 | 83.82 | 82.51 | 77.98 | 3.26 |
| 13 | 86.85 | 83.50 | 82.58 | 78.38 | 3.35 |
| 15 | 87.03 | 84.58 | 82.48 | 78.42 | 2.45 |
| 17 | 86.62 | 84.04 | 82.39 | 78.08 | 2.58 |
| 19 | 86.67 | 84.15 | 82.10 | 78.12 | 2.52 |
| 21 | 86.62 | 83.50 | 82.01 | 77.55 | 3.12 |
| 25 | 86.35 | 83.28 | 82.01 | 77.45 | 3.07 |
| 27 | 86.21 | 83.60 | 81.96 | 77.51 | 2.61 |
| 31 | 85.76 | 83.17 | 81.30 | 77.15 | 2.59 |
| 33 | 85.53 | 82.84 | 81.14 | 77.28 | 2.69 |
| 37 | 84.94 | 82.41 | 80.66 | 77.28 | 2.53 |
| 39 | 85.12 | 82.41 | 80.89 | 77.58 | 2.71 |
| 43 | 85.12 | 81.98 | 80.87 | 77.31 | 3.14 |
| 45 | 85.21 | 81.87 | 81.01 | 77.28 | 3.34 |
| 49 | 84.71 | 81.32 | 80.85 | 77.31 | 3.39 |
| 51 | 84.62 | 80.46 | 80.64 | 76.98 | 4.16 |
| 55 | 84.08 | 80.13 | 80.44 | 77.05 | 3.95 |
| 59 | 83.58 | 80.56 | 80.28 | 77.28 | 3.02 |
| 61 | 83.62 | 80.46 | 80.19 | 76.88 | 3.16 |
| 65 | 83.17 | 80.78 | 79.94 | 76.91 | 2.39 |
| 67 | 82.89 | 80.78 | 79.91 | 76.85 | 2.11 |
| 71 | 82.53 | 80.13 | 79.69 | 76.78 | 2.40 |
| 73 | 82.30 | 80.02 | 79.66 | 76.65 | 2.28 |
| 77 | 82.07 | 80.46 | 79.53 | 76.95 | 1.61 |
| 79 | 82.03 | 80.02 | 79.46 | 76.98 | 2.01 |
| 83 | 81.80 | 80.67 | 79.48 | 77.15 | 1.13 |
| 85 | 81.80 | 80.67 | 79.48 | 77.15 | 1.13 |
| 89 | 81.76 | 80.46 | 79.44 | 77.01 | 1.30 |
| 91 | 81.76 | 79.70 | 79.57 | 76.65 | 2.06 |
| 95 | 81.39 | 80.13 | 79.41 | 76.75 | 1.26 |
| 97 | 81.44 | 80.24 | 79.23 | 76.85 | 1.20 |
| 101 | 81.30 | 80.02 | 79.44 | 76.88 | 1.28 |

مشاهده می شودکه تابع فاصله اقلیدسی Recall کمتری نسبت به تابع فاصله منهتن دارد ولی بهترین نقطه برای این تابع فاصله تعداد همسایگان برابر با 15 است

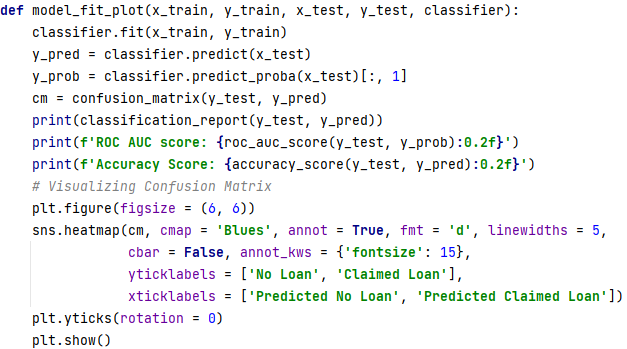
### بهترین هایپرپارامترها و رسم نمودار نتایج بهینه (رفیعی فرد)

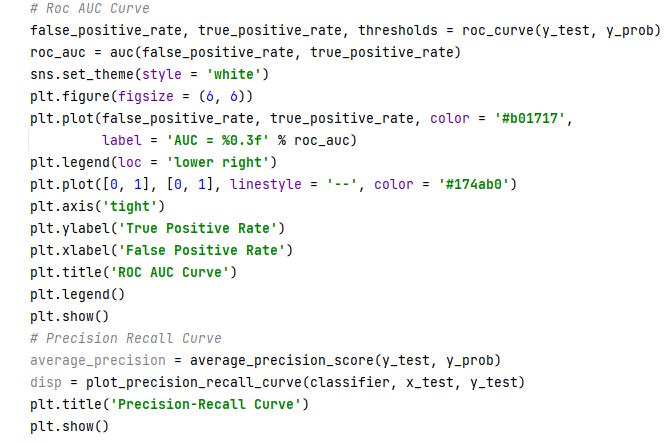
با توجه به نتایج حاصل از جستجوی گرید در فضای هایپرپارامترها و نمودارهای بالا بهترین هایپرپارامترها به شرح زیر هستند:

**metric**: manhattan

**n\_neighbors:** 19

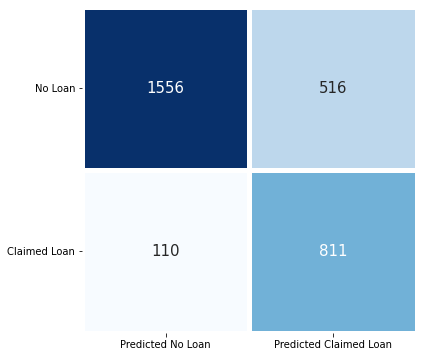
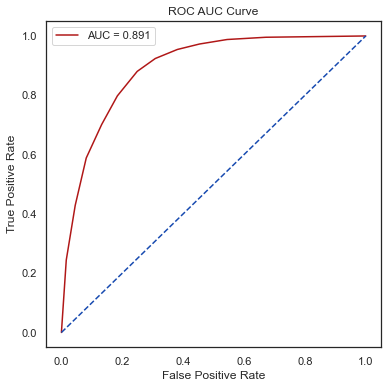
جهت رسم ماتریس آشفتگی، نمودار ROC AUC و نمودار Precision بر حسب Recall از کد زیر استفاده شد:

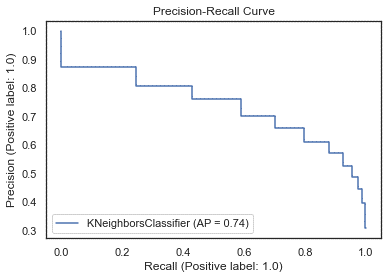




نتایج به صورت زیر هستند:

--------- Result **for** KNN on test data ---------  
  
 precision recall f1-score support  
  
 0.0 0.93 0.75 0.83 2072  
 1.0 **0.61 0.88 0.72** 921  
  
 accuracy **0.79** 2993  
 macro avg 0.77 0.82 0.78 2993  
weighted avg 0.83 0.79 0.80 2993  
  
ROC AUC score: **0.89**  
Accuracy Score: 0.79



## **ساخت مدل با SVM و تنظیم هایپرپارامترها** (رفیعی فرد)

متد ساخت مدل:

**classifier = SVC(C, kernel, random\_state)**

### معرفی هایپرپارمترها (رفیعی فرد)

1. **C** تابع هزینه
2. **kernel** نوع هسته مورد استفاده در الگوریتم

### تنظیم پارامترها برای به دست آوردن بهترین حالت بر حسب معیار Recall (رفیعی فرد)

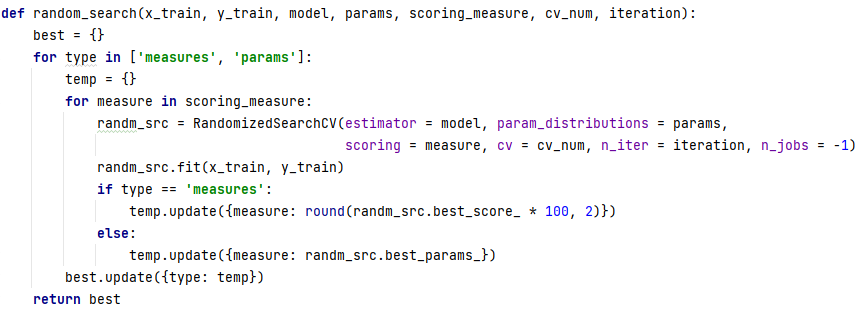
در ادامه بازه تغییر هایپرپارامترها برای استفاده از متد RandomizedSearchCV مشخص شد:

random\_params\_ab=[{  
**'algorithm'**:[**'SAMME'**, **'SAMME.R'**],  
**'learning\_rate'**:list(np.arange(0.05,0.99,0.2)),  
**'n\_estimators'**:list(np.arange(20,410,10)),  
**'random\_state'**:[20]}]

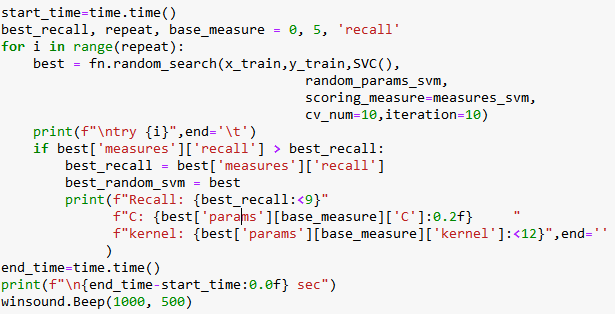
توسط کد زیر معیارهای مورد نظر را مشخص می کنیم:

measures\_svm = [**'recall'**,**'accuracy'**]

تابع زیر جهت یافتن حالت بهینه بر اساس معیار مورد نظر استفاده شد:



با استفاده از کد زیر با 5 بار اجرای تابع فوق بهترین حالت بر حسب معیار recall به دست آمد:



نتایج به شرح زیر می باشد:

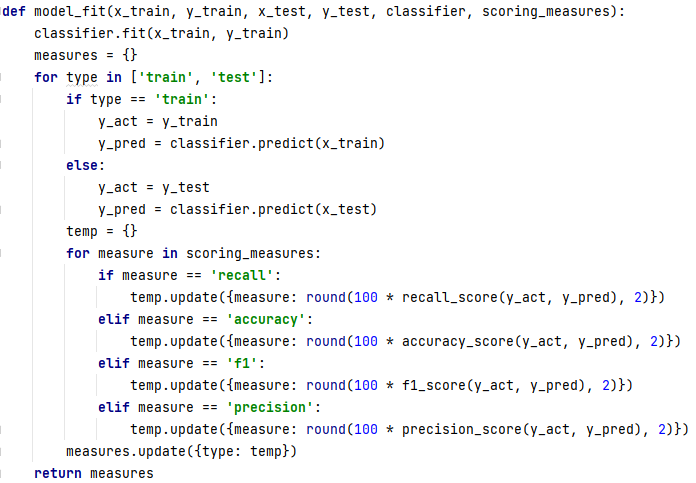
**try** 0 Recall: 88.95 C: 0.01 kernel: poly   
**try** 1   
**try** 2 Recall: 89.76 C: 0.14 kernel: poly   
**try** 3   
**try** 4   
690 sec

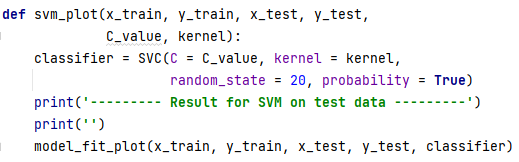
**'measures'**: {**'recall'**: 89.76, **'accuracy'**: 84.58},  
**'params'**:

**'recall'**: **'random\_state'**: 20, **'kernel'**: **'poly'**, **'C'**: 0.14},  
 **'accuracy'**: {**'random\_state'**: 20, **'kernel'**: **'linear'**, **'C'**: 0.25}}

### ارزیابی مدل بر روی داده آموزشی و تست (رفیعی فرد)

دو تابع زیر برای ارزیابی مدل استفاده شد:



****

سپس با کد زیر مقدار معیار recall برای داده آموزشی و تست محاسبه شد:

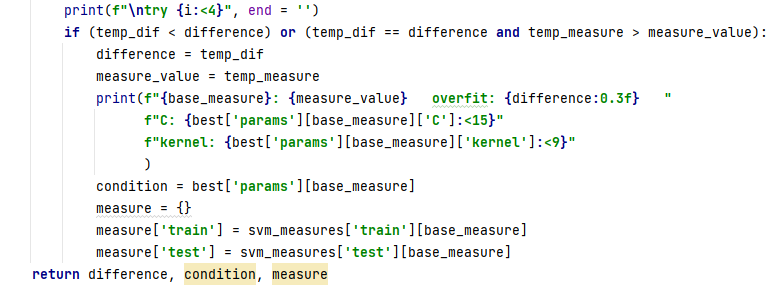
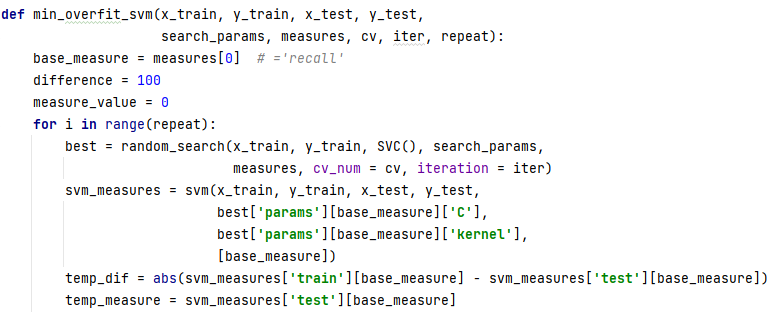


نتایج به صورت زیر به دست آمد:

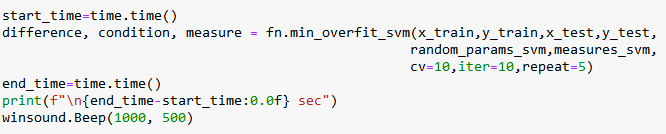
**'train'**: {**'recall'**: 89.72, **'accuracy'**: 84.14},  
**'test'**: {**'recall'**: 90.88, **'accuracy'**: 81.36}

### جستجوی کمترین حالت overfitting (رفیعی فرد)

تابع زیر برای یافتن کمترین اختلاف معیار Recall بین داده آموزشی و تست استفاده شد:



سپس با استفاده از کد زیر، حالتی را که کمترین overfitting اتفاق بیفتد به دست آمد:



نتایج به شرح زیر است:

**try** 0 recall: 90.99 underfit: 1.320 C: 0.36 kernel: rbf   
**try** 1 recall: 90.34 underfit: 0.710 C: 0.6 kernel: rbf   
**try** 2   
**try** 3 recall: 90.88 underfit: 0.480 C: 0.01 kernel: poly   
**try** 4   
681 sec

در هر بار سعی، اگر نتایج بهتر شده باشد آن را چاپ می کند. مقادیر Recall روی داده تست هستند. بهترین نتیجه هایلایت شده است.

با استفاده از دستور زیر نتایج برای داده آموزشی و تست بر اساس معیار Recall نمایش داده می شود:

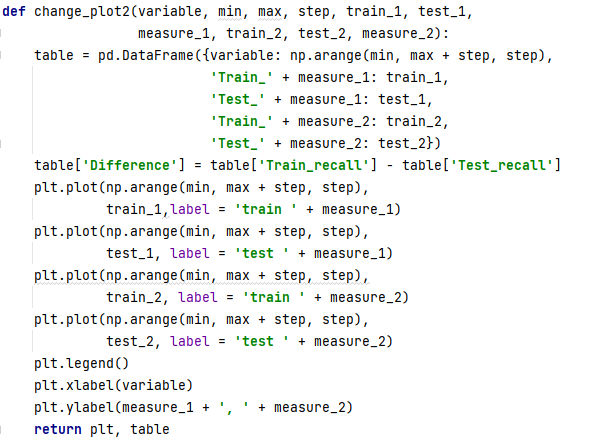


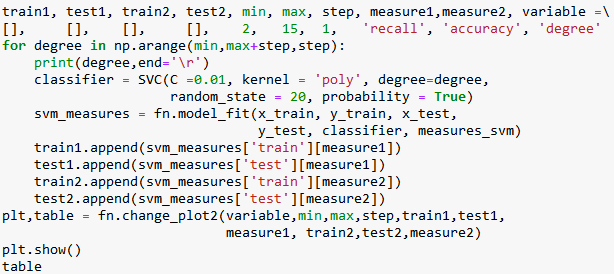
نتایج به شرح زیر است:

**'train'**: {**'recall'**: 89.67, **'accuracy'**: 83.89},  
**'test'**: {**'recall'**: 90.99, **'accuracy'**: 81.02}

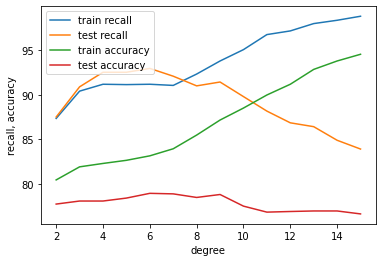
### بررسی تغییر نتایج با تغییر هایپرپارامترها (رفیعی فرد)

با استفاده از کد زیر نمودار تغییرات Recall برای داده آموزشی و تست بر حسب درجه چند جمله ای رسم شد:





خروجی کد بالا به صورت نمودار و داده جدولی به صورت زیر است:



|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| degree | Train\_recall | Test\_recall | Train\_accuracy | Test\_accuracy | Difference |
| 2 | 87.35 | 87.51 | 80.48 | 77.78 | -0.16 |
| 3 | 90.40 | 90.88 | 81.94 | 78.12 | -0.48 |
| 4 | 91.17 | 92.51 | 82.32 | 78.12 | -1.34 |
| 5 | 91.13 | 92.51 | 82.67 | 78.45 | -1.38 |
| 6 | 91.17 | 92.94 | 83.17 | 78.98 | -1.77 |
| 7 | 91.04 | 92.07 | 83.96 | 78.92 | -1.03 |
| 8 | 92.31 | 90.99 | 85.49 | 78.52 | 1.32 |
| 9 | 93.77 | 91.42 | 87.17 | 78.85 | 2.35 |
| 10 | 95.04 | 89.79 | 88.51 | 77.55 | 5.25 |
| 11 | 96.72 | 88.17 | 89.97 | 76.88 | 8.55 |
| 12 | 97.13 | 86.86 | 91.17 | 76.95 | 10.27 |
| 13 | 97.95 | 86.43 | 92.83 | 77.01 | 11.52 |
| 14 | 98.32 | 84.91 | 93.77 | 77.01 | 13.41 |
| 15 | 98.77 | 83.93 | 94.52 | 76.68 | 14.84 |

همانطور که مشاهده می شود با افزایش پیچیدگی چند جمله ای مدل روی داده آموزشی عملکرد بهتری دارد ولی روی داده تست عملکرد رو به کاهش دارد. مدل ایجاد شده با چند جمله ای درجه 6 بهترین عملکرد را روی داده تست دارد گرچه 1.77 دارای underfitting است.

### بهترین هایپرپارامترها و رسم نمودار نتایج بهینه (رفیعی فرد)

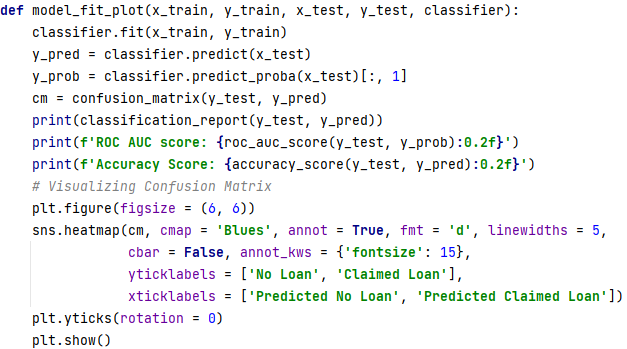
با توجه به نتایج حاصل از جستجوی تصادفی در فضای هایپرپارامترها و نمودارهای بالا بهترین هایپرپارامترها به شرح زیر هستند:

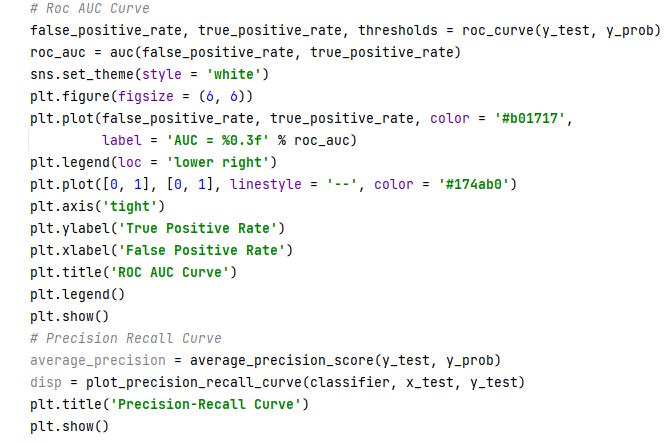
**C**: 0.01

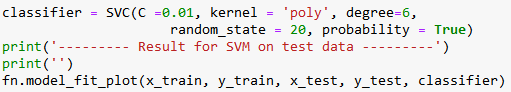
**kernel:** poly

**degree:** 6

جهت رسم ماتریس آشفتگی، نمودار ROC AUC و نمودار Precision بر حسب Recall از کد زیر استفاده شد:

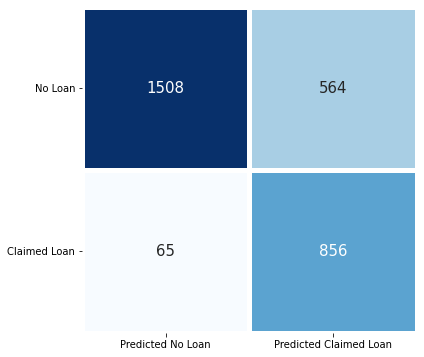
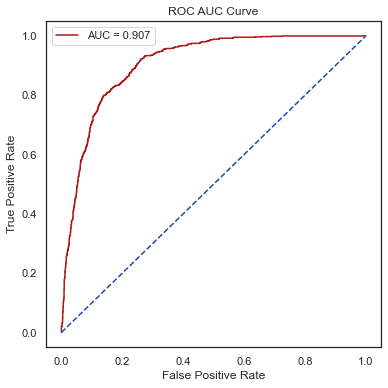


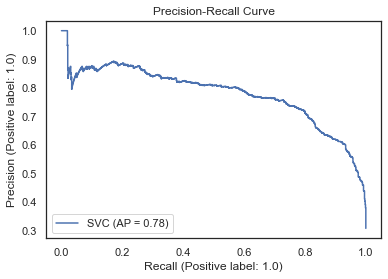




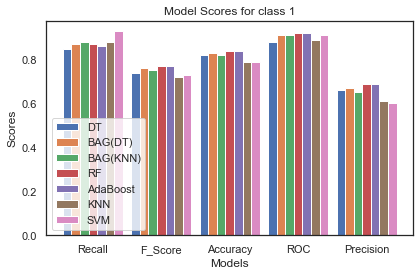
نتایج به صورت زیر هستند:

--------- Result **for** SVM on test data ---------  
  
 precision recall f1-score support  
  
 0.0 0.96 0.73 0.83 2072  
 1.0 **0.60 0.93 0.73** 921  
  
 accuracy **0.79** 2993  
 macro avg 0.78 0.83 0.78 2993  
weighted avg 0.85 0.79 0.80 2993  
  
ROC AUC score: **0.91**  
Accuracy Score: 0.79



# مقایسه دسته بندها(رفیعی فرد-سلامی)



# تفسیر نتایج(رفیعی فرد)

مدل SVM با کرنل چند جمله ای درجه 6 و Recall برابر با 92.94% روی داده تست بهترین دسته بند است.

درخت تصمیم پایین ترین مقدار Recall را دارد

بعد از SVM دسته بندهای KNN و Bagging با دسته بند پایه KNN بهترین عملکرد را دارند.

این نتایج نشان دهنده این است که داده ها به صورت خطی تفکیک پذیر نیستند و پایین بودن عملکرد درخت تصمیم هم گواه همین موضوع است و اینکه داده ها با یک چند جمله ای درجه 6 بهترین تفکیک پذیری را دارند نیز گواه دیگری بر این موضوع است.