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](#_Toc126855677)

[1-1 مجموعه داده در یک نگاه (رفیعی فرد) 5](#_Toc126855678)

[1-2 متغیرهای ورودی (رفیعی فرد) 6](#_Toc126855679)

[1-3 متغیر خروجی (رفیعی فرد) 20](#_Toc126855680)

[**2** فراخوانی کتابخانه های مورد نیاز و انجام تنظیمات **pyspark** 21](#_Toc126855681)

[3 تجمیع چانک های مجموعه داده 21](#_Toc126855682)

[4 تحلیل اکتشافی داده (**Exploratory data analysis**) 22](#_Toc126855683)

[5 مرحله پیش پردازش 22](#_Toc126855684)

[5-1 تبدیل به داده‌های عددی 22](#_Toc126855685)

[5-2 نرمال سازی 24](#_Toc126855686)

[5-3 تشخیص و حذف داده های پرت 24](#_Toc126855687)

[5-4 بررسی مقادیر از دست رفته (**Missing Values**) 24](#_Toc126855688)

[5-4-1 ایجاد مقادیر از دست رفته مصنوعی (سلامی) 25](#_Toc126855689)

[5-4-2 پر کردن مقادیر از دست رفته و بررسی آن‌ها (رفیعی فرد-سلامی) 25](#_Toc126855690)

[5-5 انتخاب متغیر‌ها (سلامی) 26](#_Toc126855691)

[5-5-1 انتخاب متغیر با random forest (سلامی) 26](#_Toc126855692)

[5-5-2 انتخاب متغیر با Boruta (رفیعی فرد) 27](#_Toc126855693)

[6 ساخت مدل (رفیعی فرد) 28](#_Toc126855694)

[6-1 آماده سازی داده آموزشی و تست (رفیعی فرد) 28](#_Toc126855695)

[6-2 معیار اصلی جهت انتخاب بهترین مدل (رفیعی فرد) 28](#_Toc126855696)

[6-3 ساخت مدل با درخت تصمیم و تنظیم هایپرپارامترهای آن (رفیعی فرد) 31](#_Toc126855697)

[6-3-1 معرفی هایپرپارمترها (رفیعی فرد) 31](#_Toc126855698)

[6-3-2 تنظیم پارامترها برای به دست آوردن بهترین حالت بر حسب معیار Recall (رفیعی فرد) 31](#_Toc126855699)

[6-3-3 ارزیابی مدل بر روی داده آموزشی و تست (رفیعی فرد) 32](#_Toc126855700)

[6-3-4 جستجوی کمترین حالت overfitting (رفیعی فرد) 33](#_Toc126855701)

[6-3-5 بررسی تغییر نتایج با تغییر هایپرپارامترها (رفیعی فرد) 34](#_Toc126855702)

[6-3-6 بهترین هایپرپارامترها و رسم نمودار نتایج بهینه (رفیعی فرد) 36](#_Toc126855703)

[6-4 ساخت مدل با **Bagging** و مدل پایه درخت تصمیم و تنظیم هایپرپارامترهای آن (رفیعی فرد) 39](#_Toc126855704)

[6-4-1 معرفی هایپرپارمترها (رفیعی فرد) 39](#_Toc126855705)

[6-4-2 تنظیم پارامترها برای به دست آوردن بهترین حالت بر حسب معیار Recall (رفیعی فرد) 39](#_Toc126855706)

[6-4-3 ارزیابی مدل بر روی داده آموزشی و تست (رفیعی فرد) 40](#_Toc126855707)

[6-4-4 جستجوی کمترین حالت overfitting (رفیعی فرد) 41](#_Toc126855708)

[6-4-5 بررسی تغییر نتایج با تغییر هایپرپارامترها (رفیعی فرد) 43](#_Toc126855709)

[6-4-6 بهترین هایپرپارامترها و رسم نمودار نتایج بهینه (رفیعی فرد) 49](#_Toc126855710)

[6-5 **ساخت مدل با Bagging و مدل پایه KNN و تنظیم هایپرپارامترها** (رفیعی فرد) 51](#_Toc126855711)

[6-5-1 معرفی هایپرپارمترها (رفیعی فرد) 51](#_Toc126855712)

[6-5-2 تنظیم پارامترها برای به دست آوردن بهترین حالت بر حسب معیار Recall (رفیعی فرد) 52](#_Toc126855713)

[6-5-3 ارزیابی مدل بر روی داده آموزشی و تست (رفیعی فرد) 53](#_Toc126855714)

[6-5-4 جستجوی کمترین حالت overfitting (رفیعی فرد) 54](#_Toc126855715)

[6-5-5 بررسی تغییر نتایج با تغییر هایپرپارامترها (رفیعی فرد) 55](#_Toc126855716)

[6-5-6 بهترین هایپرپارامترها و رسم نمودار نتایج بهینه (رفیعی فرد) 60](#_Toc126855717)

[6-6 **ساخت مدل با جنگل تصادفی و تنظیم هایپرپارامترها** (رفیعی فرد) 63](#_Toc126855718)

[6-6-1 معرفی هایپرپارمترها (رفیعی فرد) 63](#_Toc126855719)

[6-6-2 تنظیم پارامترها برای به دست آوردن بهترین حالت بر حسب معیار Recall (رفیعی فرد) 63](#_Toc126855720)

[6-6-3 ارزیابی مدل بر روی داده آموزشی و تست (رفیعی فرد) 64](#_Toc126855721)

[6-6-4 جستجوی کمترین حالت overfitting (رفیعی فرد) 66](#_Toc126855722)

[6-6-5 بررسی تغییر نتایج با تغییر هایپرپارامترها (رفیعی فرد) 67](#_Toc126855723)

[6-6-6 بهترین هایپرپارامترها و رسم نمودار نتایج بهینه (رفیعی فرد) 69](#_Toc126855724)

[6-7 **ساخت مدل با آدابوست و تنظیم هایپرپارامترها** (رفیعی فرد) 72](#_Toc126855725)

[6-7-1 معرفی هایپرپارمترها (رفیعی فرد) 72](#_Toc126855726)

[6-7-2 تنظیم پارامترها برای به دست آوردن بهترین حالت بر حسب معیار Recall (رفیعی فرد) 72](#_Toc126855727)

[6-7-3 ارزیابی مدل بر روی داده آموزشی و تست (رفیعی فرد) 73](#_Toc126855728)

[6-7-4 جستجوی کمترین حالت overfitting (رفیعی فرد) 74](#_Toc126855729)

[6-7-5 بررسی تغییر نتایج با تغییر هایپرپارامترها (رفیعی فرد) 76](#_Toc126855730)

[6-7-6 بهترین هایپرپارامترها و رسم نمودار نتایج بهینه (رفیعی فرد) 80](#_Toc126855731)

[6-8 **ساخت مدل با KNN و تنظیم هایپرپارامترها** (رفیعی فرد) 83](#_Toc126855732)

[6-8-1 معرفی هایپرپارمترها (رفیعی فرد) 83](#_Toc126855733)

[6-8-2 تنظیم پارامترها برای به دست آوردن بهترین حالت بر حسب معیار Recall (رفیعی فرد) 83](#_Toc126855734)

[6-8-3 ارزیابی مدل بر روی داده آموزشی و تست (رفیعی فرد) 84](#_Toc126855735)

[6-8-4 بررسی تغییر نتایج با تغییر هایپرپارامترها (رفیعی فرد) 85](#_Toc126855736)

[6-8-5 بهترین هایپرپارامترها و رسم نمودار نتایج بهینه (رفیعی فرد) 88](#_Toc126855737)

[6-9 **ساخت مدل با SVM و تنظیم هایپرپارامترها** (رفیعی فرد) 90](#_Toc126855738)

[6-9-1 معرفی هایپرپارمترها (رفیعی فرد) 90](#_Toc126855739)

[6-9-2 تنظیم پارامترها برای به دست آوردن بهترین حالت بر حسب معیار Recall (رفیعی فرد) 90](#_Toc126855740)

[6-9-3 ارزیابی مدل بر روی داده آموزشی و تست (رفیعی فرد) 91](#_Toc126855741)

[6-9-4 جستجوی کمترین حالت overfitting (رفیعی فرد) 92](#_Toc126855742)

[6-9-5 بررسی تغییر نتایج با تغییر هایپرپارامترها (رفیعی فرد) 94](#_Toc126855743)

[6-9-6 بهترین هایپرپارامترها و رسم نمودار نتایج بهینه (رفیعی فرد) 95](#_Toc126855744)

[7 مقایسه دسته بندها(رفیعی فرد-سلامی) 98](#_Toc126855745)

[8 تفسیر نتایج(رفیعی فرد) 98](#_Toc126855746)

# 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:

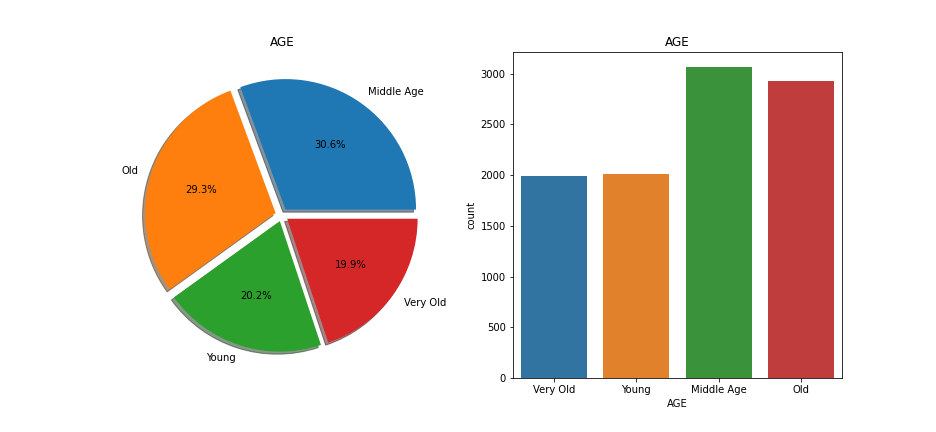
['16-25', '26-39', '40-64', '65+']

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:

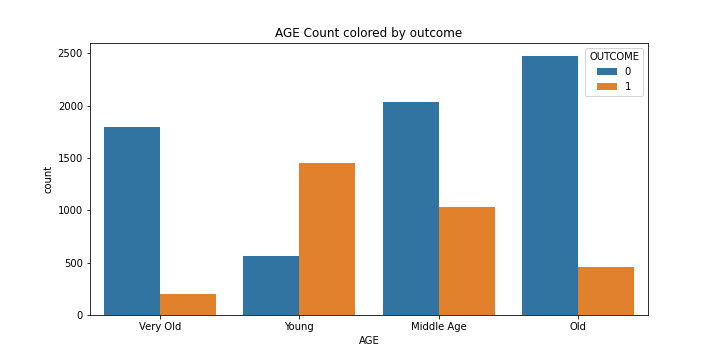
 Middle Age 3063

Old 2931

Young 2016

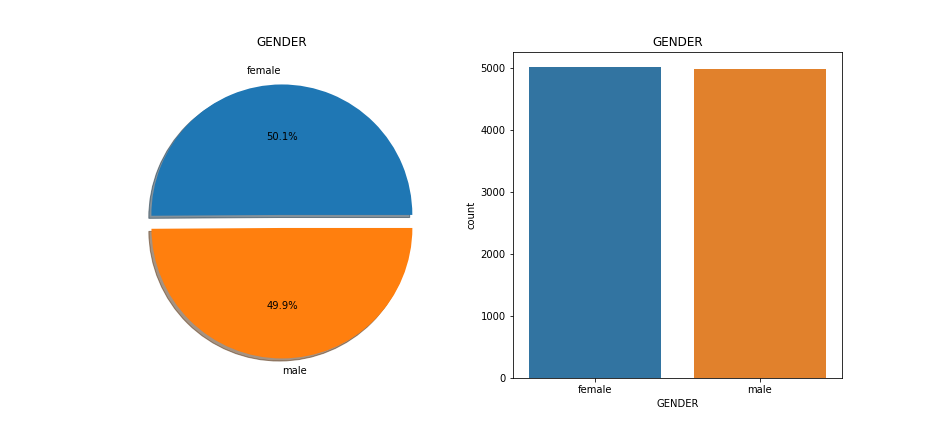
Very Old 1990

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.

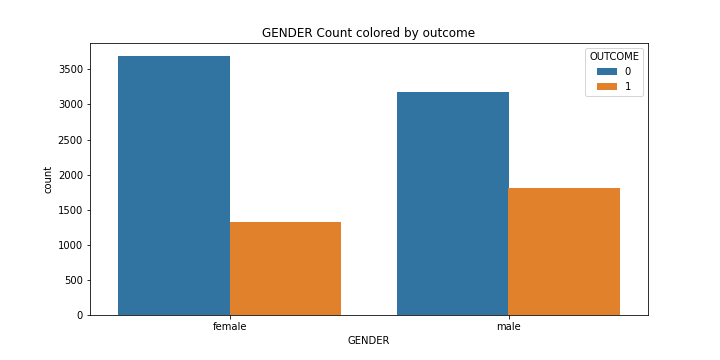
نمودار توزیع این ویژگی به صورت زیر است:



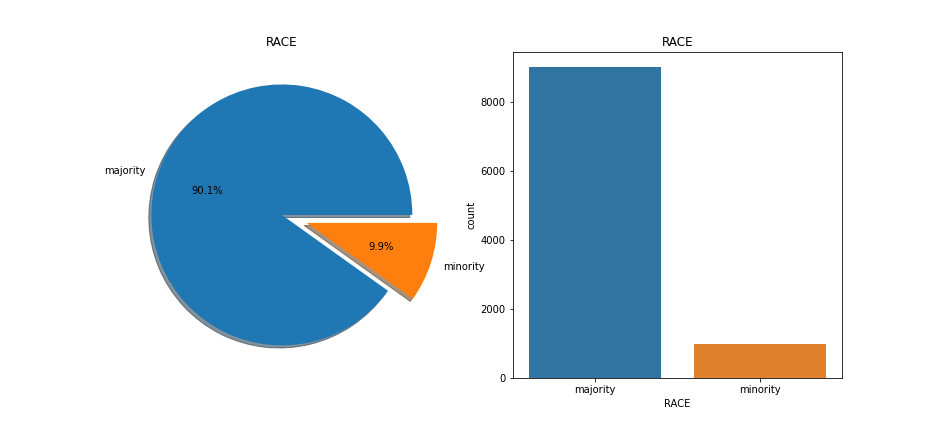
female 5010

male 4990

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

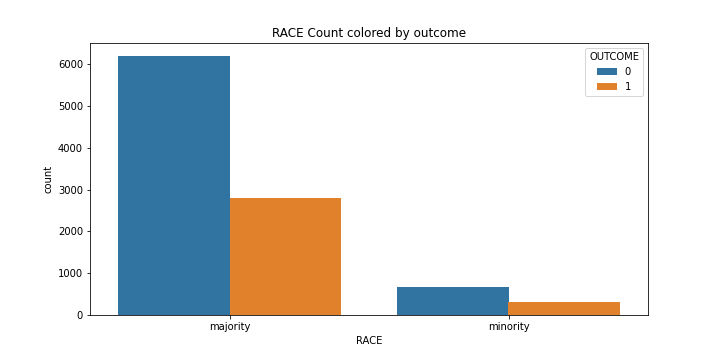


1. نژاد (RACE): از نوع باینری است که دارای مقادیر ['majority' 'minority'] به معنی اکثریت و اقلیت تقسیم می گردد و نمودار توزیع آن به صورت زیر است:

 majority 9012

minority 988

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



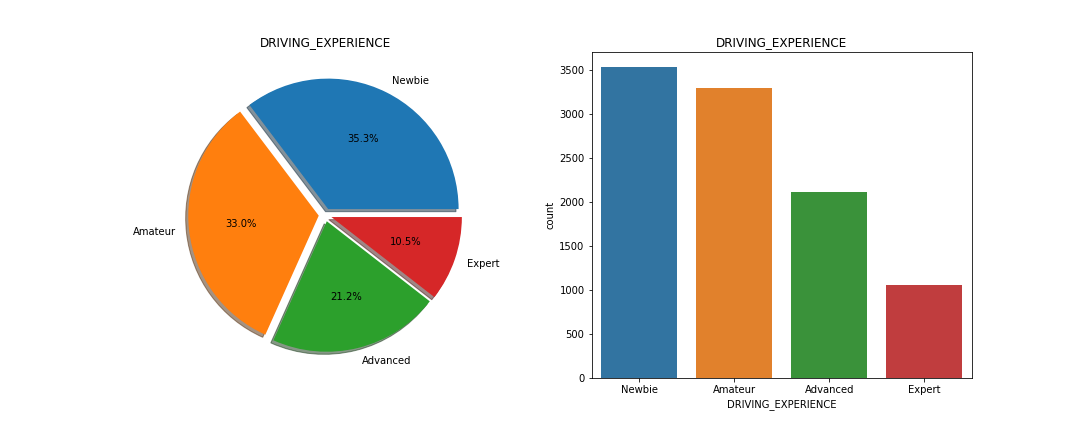
1. مهارت رانندگی (DRIVING\_EXPERIENCE): از نوع ترتیبی و در یکی از بازه های زیر است:

['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:

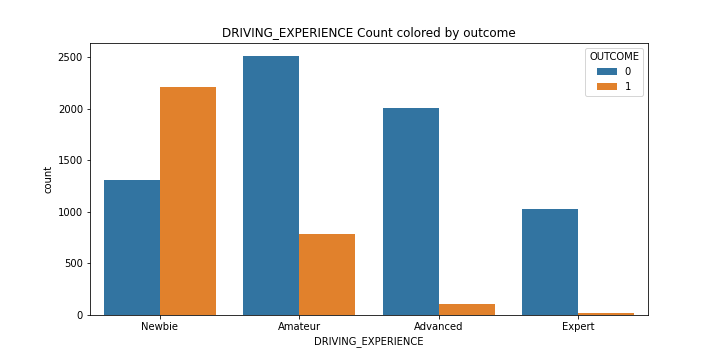
 Newbie 3530

Amateur 3299

Advanced 2119

Expert 1052

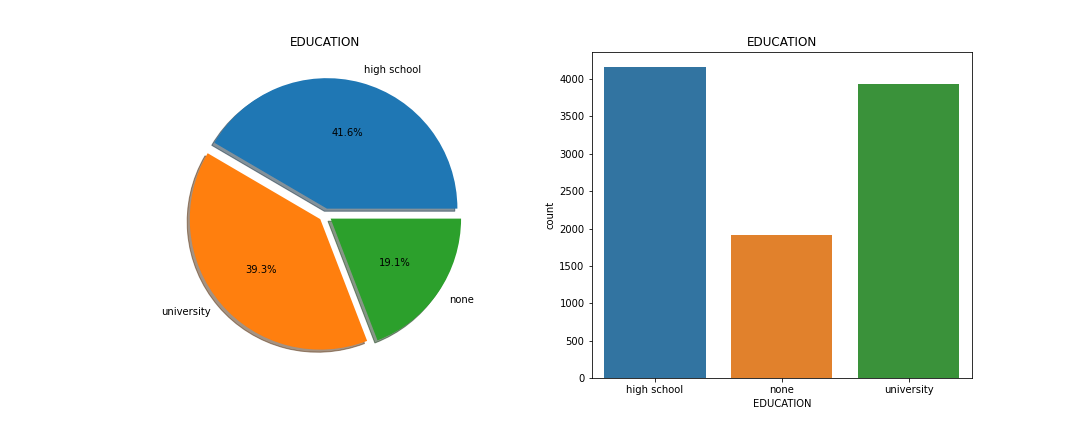
توزیع این ویژگی بر حسب دو کلاس به صورت زیر می باشد:



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

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

نمودار توزیع آن به صورت زیر است:

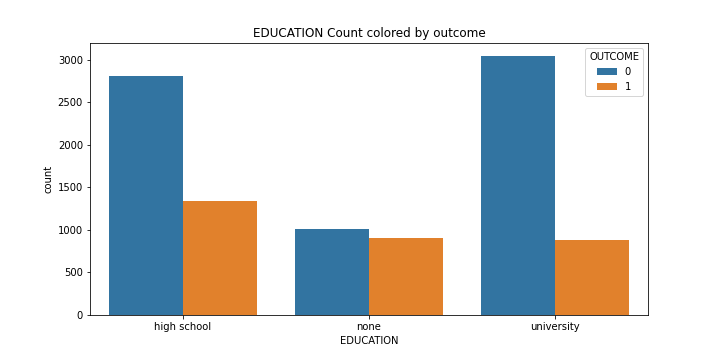


high school 4157

university 3928

none 1915

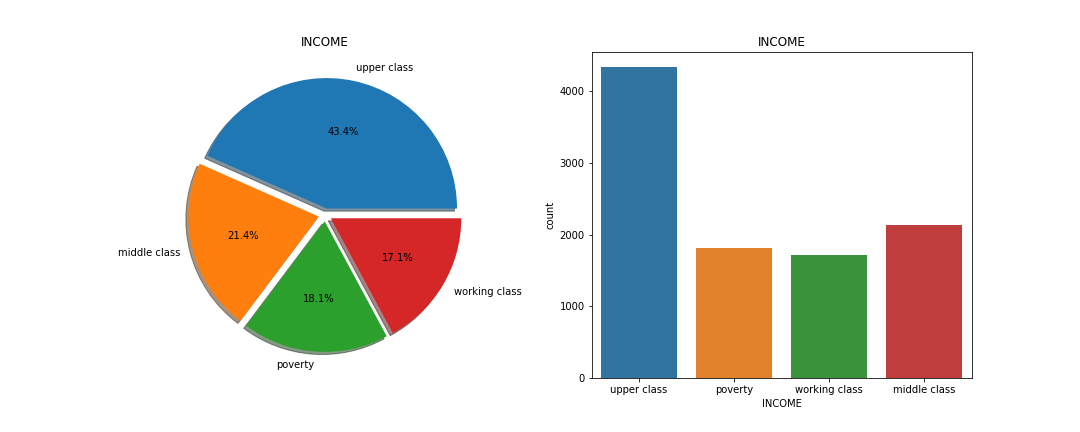
توزیع این ویژگی بر حسب دو کلاس به صورت زیر می باشد:



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:

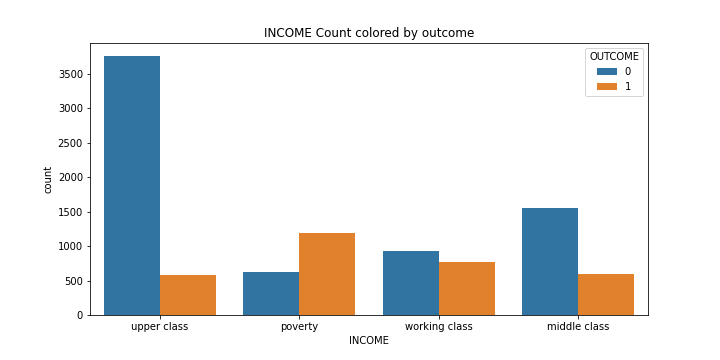
 upper class 4336

middle class 2138

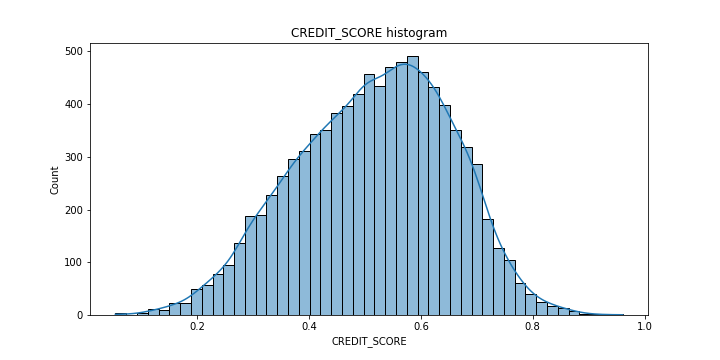
poverty 1814

working class 1712

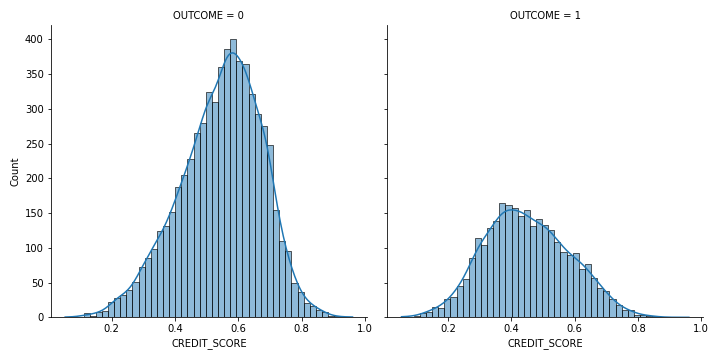
توزیع این ویژگی بر حسب دو کلاس به صورت زیر می باشد:



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:



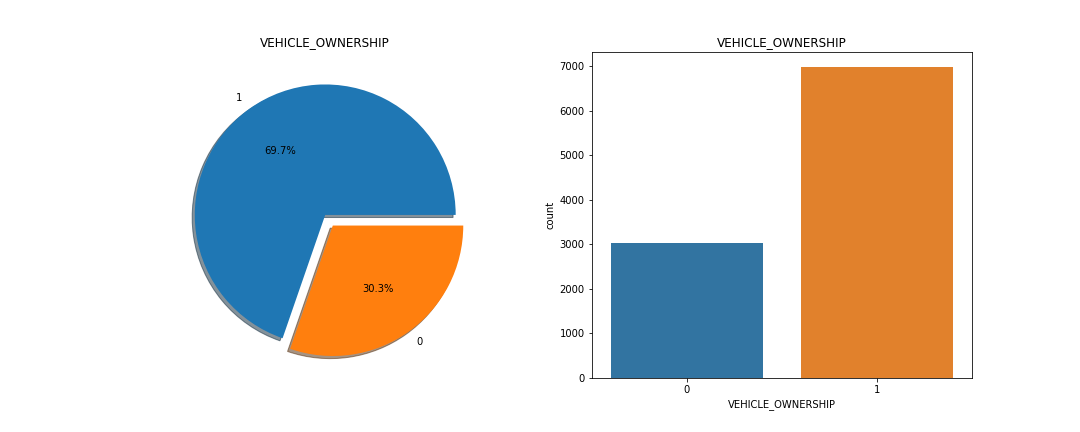
توزیع این ویژگی بر حسب دو کلاس به صورت زیر می باشد:



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.

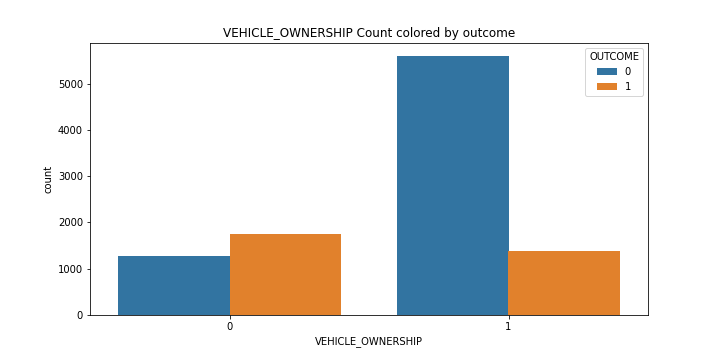
توضیح: با توجه به اینکه مجموعه داده در مورد بیمه خودرو است بنابراین مقدار 0 در این ویژگی به معنی این است که خودرو در تصرف فرد است ولی او مالک خودرو نیست مانند راننده تاکسی یا کسانی که خودرو را اجاره کرده اند یا به هر نحوی از خودرو شخص دیگری استفاده می کنند.

نمودار توزیع آن به صورت زیر است:

 1 6970

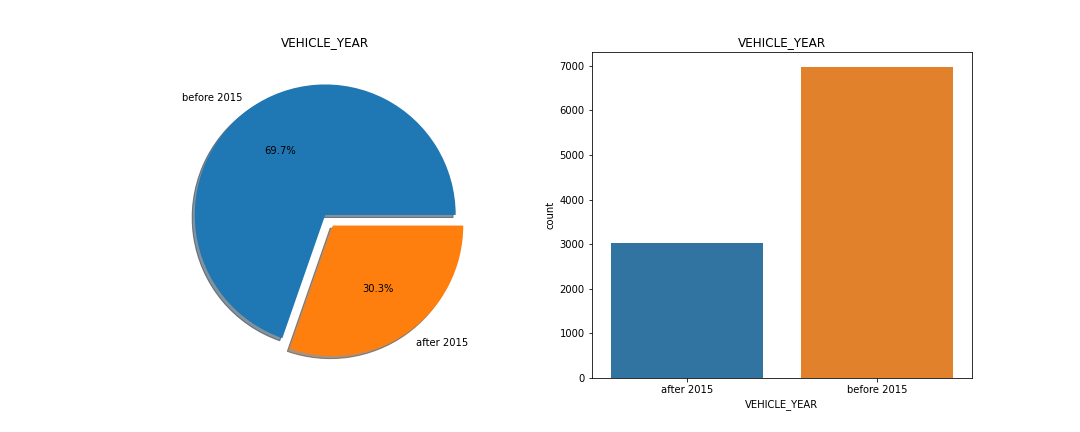
0 3030

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



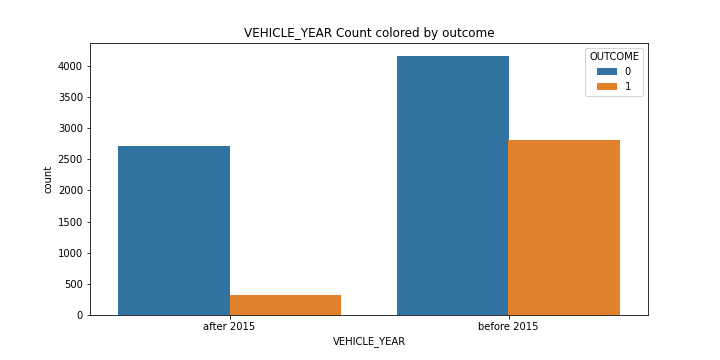
1. سال تولید خودرو (VEHICLE\_YEAR): از نوع باینری و دارای یکی از مقادیر زیر است:

[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:

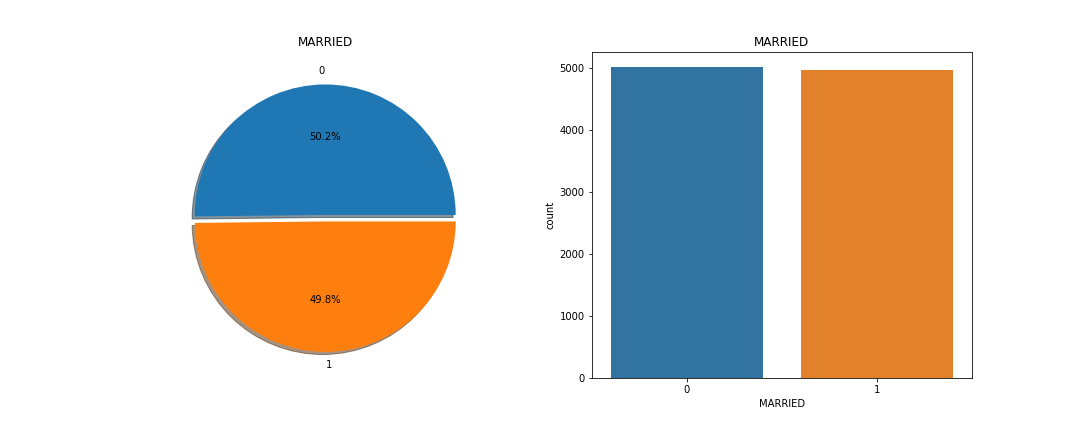
 before 2015 6967

after 2015 3033

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

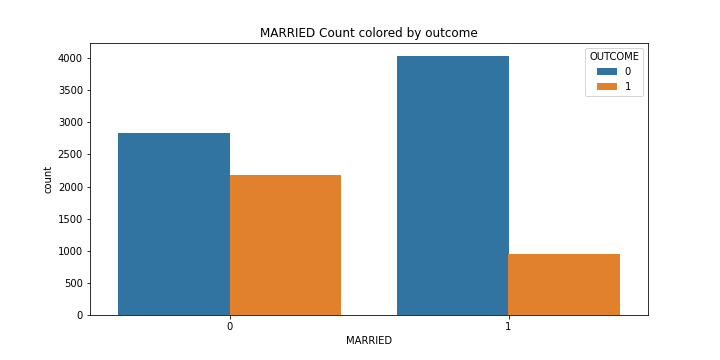


1. وضعیت تأهل (MARRIED): از نوع باینری است و دارای یکی از مقادیر 0 به معنی مجرد و 1 به معنی متأهل است و نمودار توزیع آن به صورت زیر است:

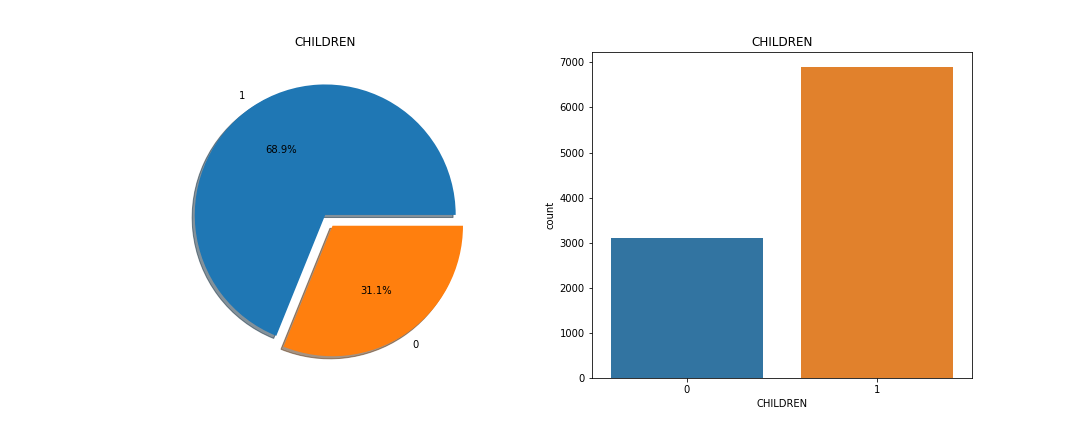
 0 5018

1 4982

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



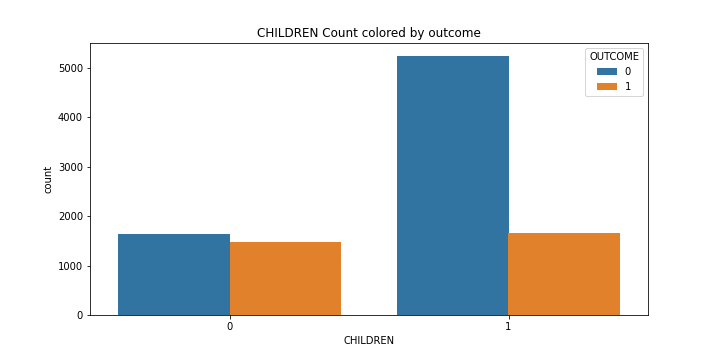
1. وضعیت فرزند (CHILDREN): از نوع باینری است و دارای یکی از مقادیر 0 به معنی عدم داشتن فرزند و 1 به معنی داشتن فرزند است و نمودار توزیع آن به صورت زیر است:



1 6888

0 3112

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



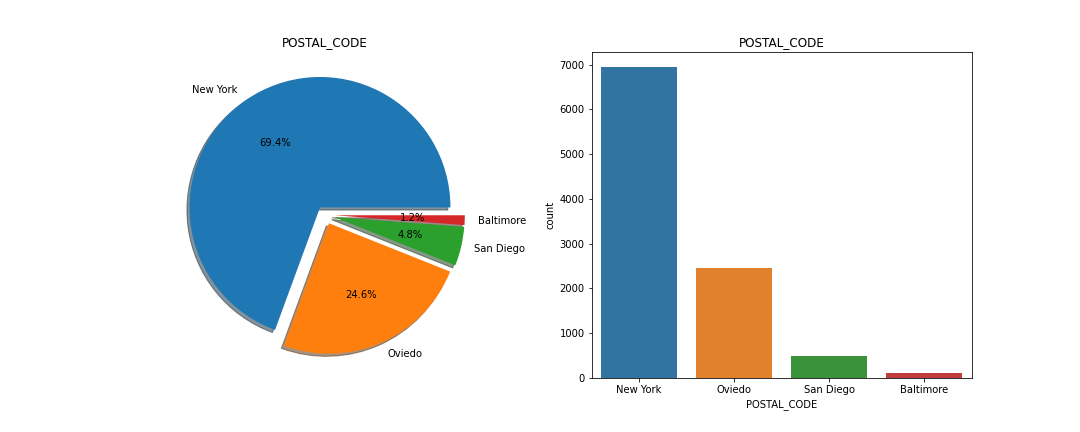
1. محل سکونت (POSTAL\_CODE): از نوع اسمی است و دارای یکی از مقادیر زیر است:

['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:

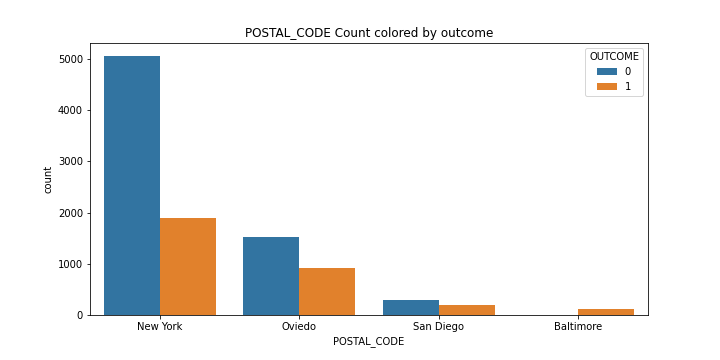
 New York 6940

Oviedo 2456

San Diego 484

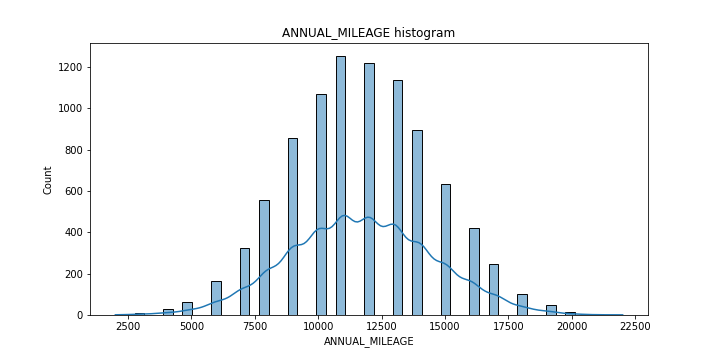
Baltimore 120

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

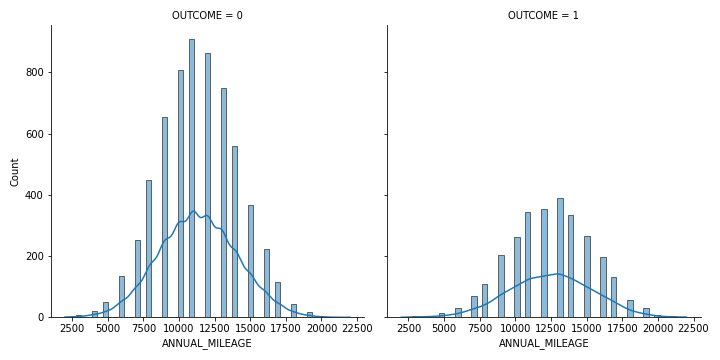


1. مسافت پیموده شده سالیانه (ANNUAL\_MILEAGE): از نوع عددی صحیح (int) و مقدار آن بین 2000 تا 22000 است.

نمودار توزیع آن به صورت زیر است:



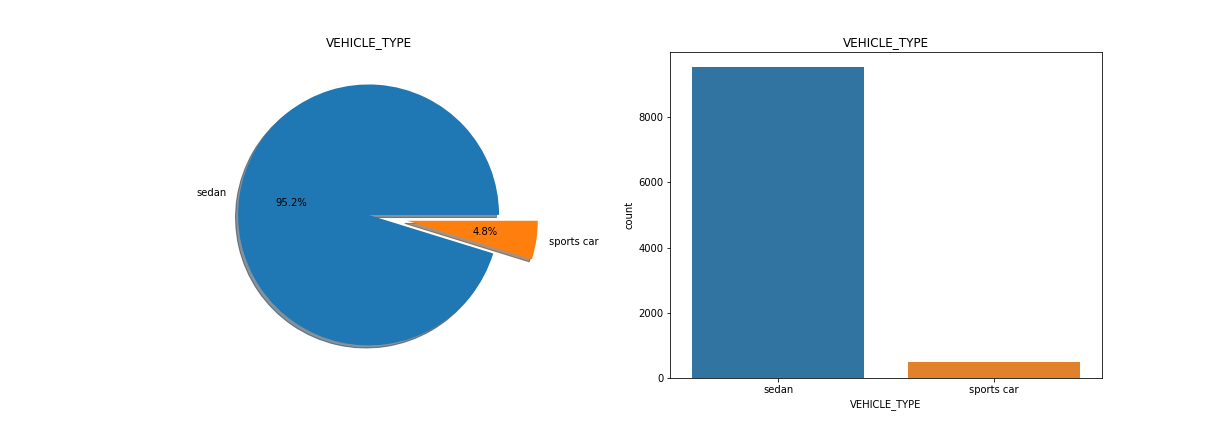
و توزیع این ویژگی بر حسب دو کلاس به صورت زیر می باشد:



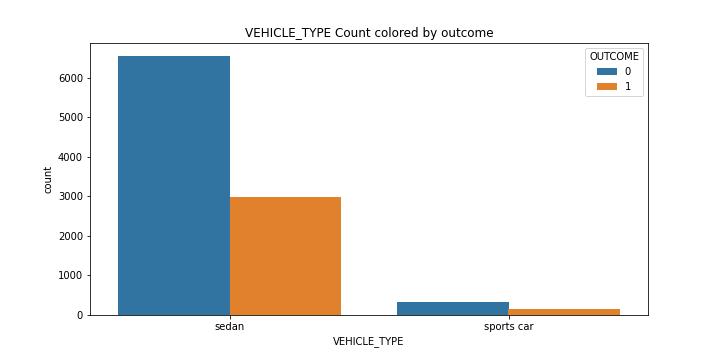
1. نوع خودرو (VEHICLE\_TYPE): از نوع باینری و دارای یکی از مقادیر زیر است:

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

نمودار توزیع آن به صورت زیر است:

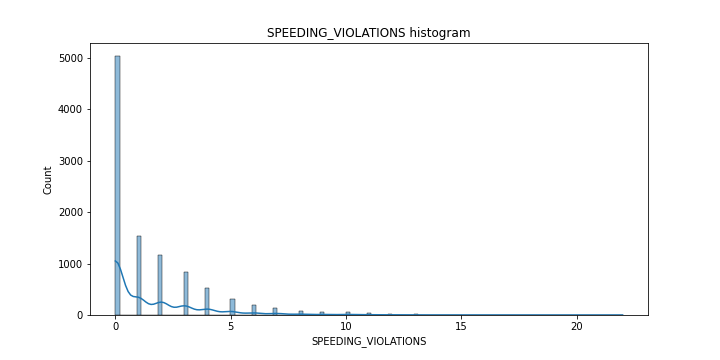


و توزیع این ویژگی بر حسب دو کلاس به صورت زیر می باشد:

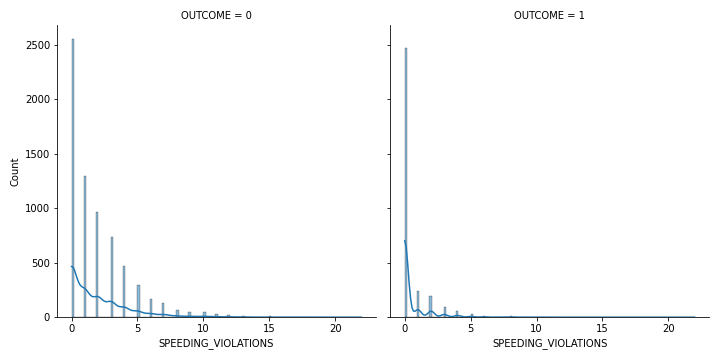


1. تخلف سرعت (SPEEDING\_VIOLATIONS): از نوع عددی صحیح (int) است که مقدار آن بین 0 و 22 تغییر می کند و بیانگر تعداد دفعات تعدی راننده از سرعت مجاز است.

نمودار توزیع آن به صورت زیر است:

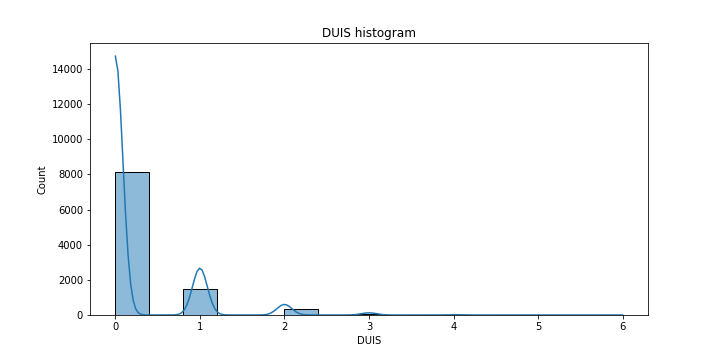


و توزیع این ویژگی بر حسب دو کلاس به صورت زیر می باشد:

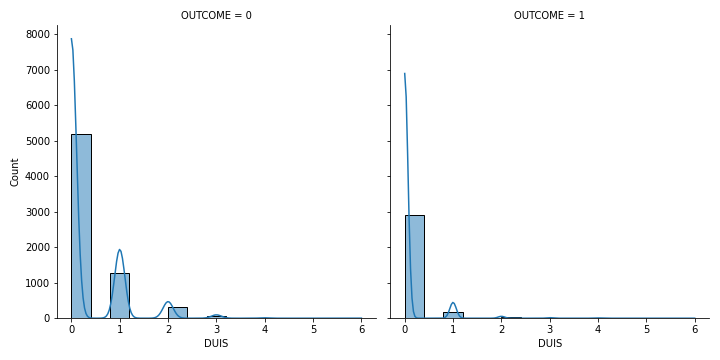


1. تخلف مصرف مشروبات الکلی (DUIS): از نوع عددی صحیح (int) و مقدار آن بین 0 تا 6 متغیر است.

نمودار توزیع آن به صورت زیر است:

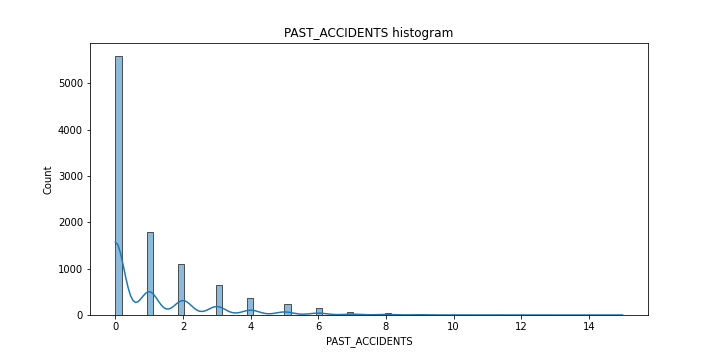


و توزیع این ویژگی بر حسب دو کلاس به صورت زیر می باشد:

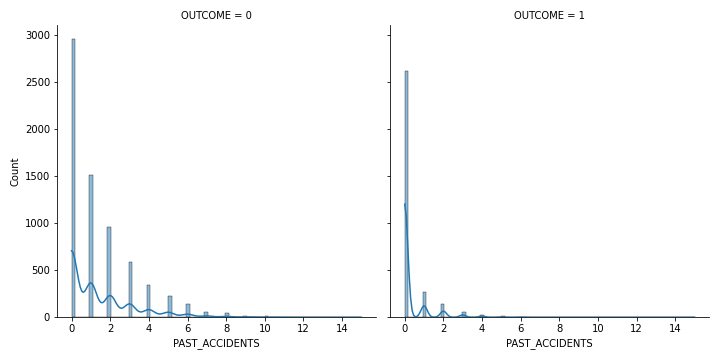


1. تعداد تصادفات گذشته (PAST\_ACCIDENTS): از نوع عددی صحیح (int) و مقدار آن بین 0 تا 15 است.

نمودار توزیع آن به صورت زیر است:

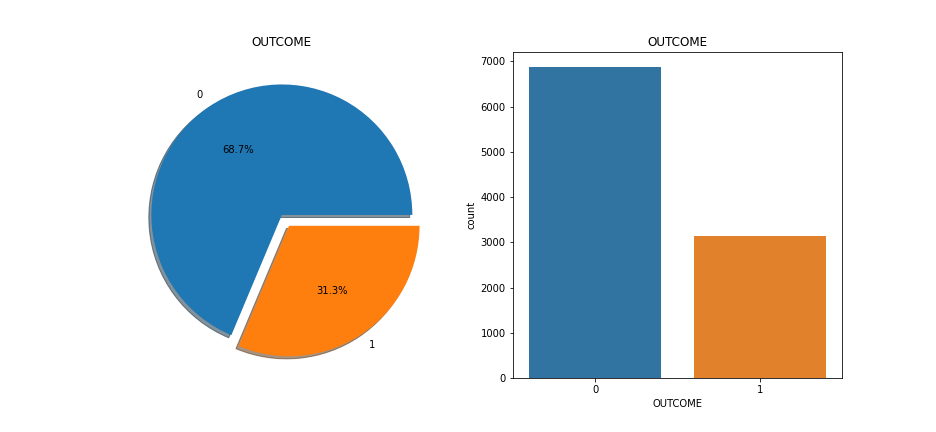


و توزیع این ویژگی بر حسب دو کلاس به صورت زیر می باشد:



## 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\

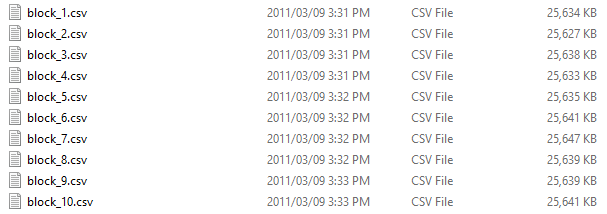
    .appName('myApp')\

        .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**)

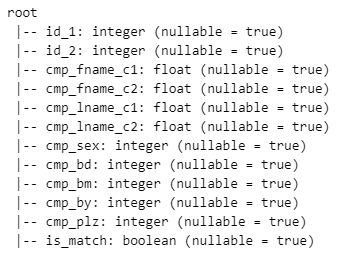
قرار هست خانم حمیدی زحمت این قسمت را بکشند 😊

# مرحله پیش پردازش

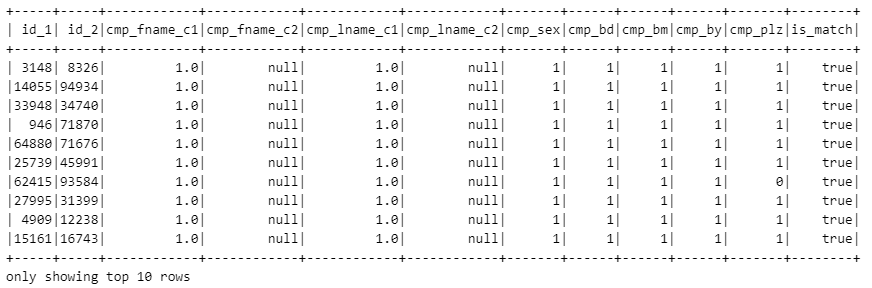
## تبدیل به داده‌های عددی

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:

from pyspark.sql.functions import when, lit

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

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

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

                                  lit(1)).otherwise(0)

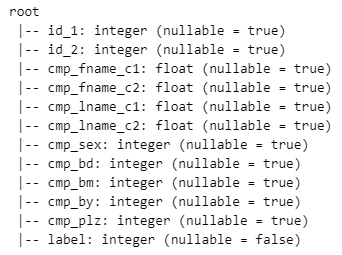
                                  )

    return temp

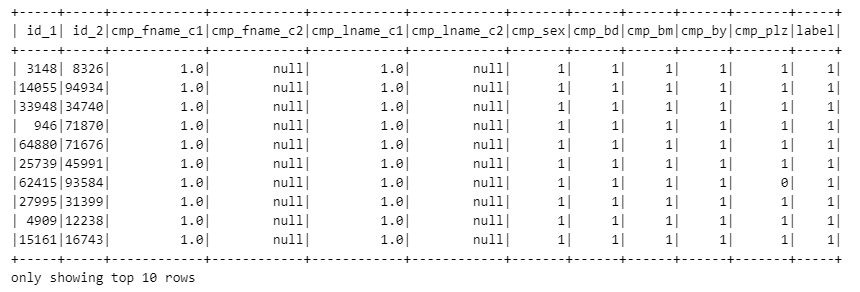
numerical\_df = convert\_label\_binary(df).drop('is\_match')

که نتیجه آن به صورت زیر می باشد:

numerical\_df.printSchema()



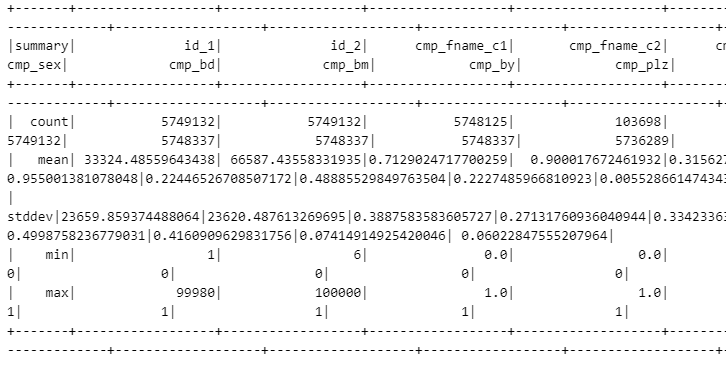
numerical\_df.show(10)



## 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.

# داده‏های از دست رفته (**Missing Values**)

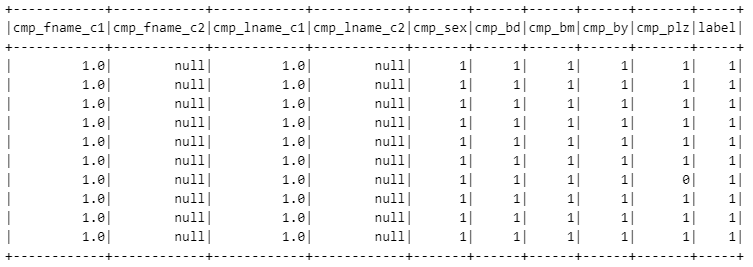
## بررسی داده‏های از دست رفته

از آنجا که ویژگی id برای هر فرد و ترکیب id\_1 و id\_2 برای هر رکورد منحصر به فرد است برای ایجاد مدل باید حذف شوند که این کار را با کد زیر انجام دادیم:

no\_id\_numerical\_df = numerical\_df.drop('id\_1','id\_2')

که نتیجه آن به صورت زیر است:

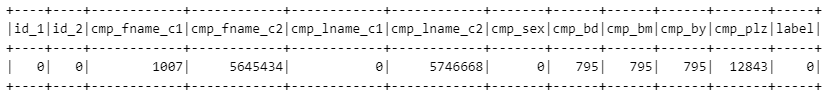
no\_id\_numerical\_df.show(10)



توسط کد زیر وضعیت داده‏های 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:



اگر رکورد های شامل همه داده های از دست رفته را حذف کنیم تنها 20 رکورد باقی می ماند.

با توجه به تعداد کل رکوردها هیچ رکوردی که همه یا حداقل 2 ویژگی آن داده‏های از دست رفته داشته باشد وجود ندارد:

تعداد کل رکوردها:

no\_id\_numerical\_df.count()



تعداد رکوردهای باقیمانده در صورت حذف رکوردهایی که تمام ویژگی های آنها Null باشد:

no\_id\_numerical\_df.na.drop(how='all').count()



تعداد رکوردهای باقیمانده در صورت حذف رکوردهایی که حداقل 2 ویژگی آنها Null باشد:

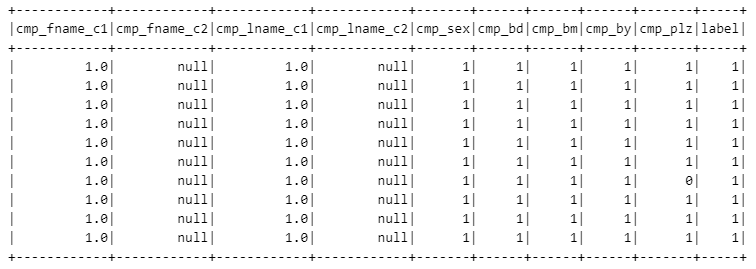
miss\_df.na.drop(how='any',thresh=2).count()



## پر کردن داده‏های از دست رفته

همانگونه که در قسمت قبل ذکر شد مجموعه داده دارای داده‏های از دست رفته زیادی مخصوصاً در ویژگی‏های cmp\_fname\_c2 و cmp\_lname\_c2 است:

no\_id\_numerical\_df.show(10)



نمونه داده قبل از پر کردن داده‏های از دست رفته

با استفاده از روش Imputer داده‏های از دست رفته را پر کردیم که کد آن به صورت زیر می باشد:

from pyspark.ml.feature import Imputer

def fill\_missing\_values(input\_df):

    # for float variables

    miss\_df=input\_df.drop('id\_1','id\_2')

    miss\_df=miss\_df.replace('?',None)

    float\_cols=[

    'cmp\_fname\_c1',

    'cmp\_fname\_c2',

    'cmp\_lname\_c1',

    'cmp\_lname\_c2',

    ]

    float\_imputer = Imputer(

        inputCols=float\_cols,

        outputCols=[f"{col}\_imputed" for col in float\_cols]

    ).setStrategy('mean')

    # for binary variables

    binary\_cols=[

        'cmp\_sex',

        'cmp\_bd',

        'cmp\_bm',

        'cmp\_by',

        'cmp\_plz',

    ]

    binary\_imputer = Imputer(

        inputCols=binary\_cols,

        outputCols=[f"{col}\_imputed" for col in binary\_cols]

    ).setStrategy('mode')

    imputed\_df=float\_imputer.fit(miss\_df).transform(miss\_df)

    output\_df=binary\_imputer.fit(imputed\_df).transform(imputed\_df)

    output\_df=output\_df.select([x for x in output\_df.columns if '\_imputed' in x or x=='is\_match'])

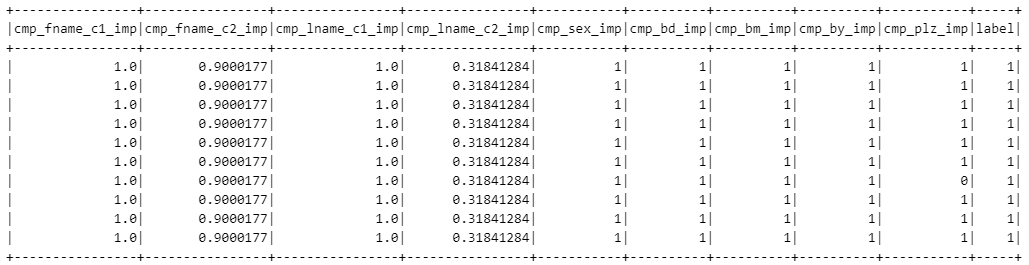
    return output\_df

def preprocessing\_df(input\_df):

    return convert\_label\_binary(fill\_missing\_values(input\_df))

prep\_df=preprocessing\_df(df)

نتیجه به صورت زیر می باشد:

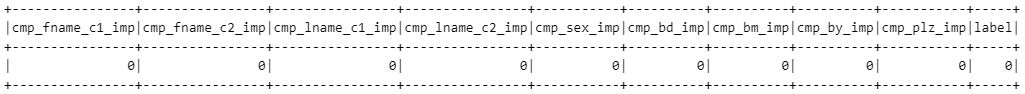


نمونه داده پس از پر کردن داده‏های از دست رفته

توسط کد زیر چک می‏کنیم که تمام داده‏های از دست رفته پر شده باشد:

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

نتیجه به صورت زیر می‏باشد:



همانطور که ملاحظه می‏شود هیچ داده از دست رفته ای برای هیچ کدام از ویژگی‏ها وجود ندارد.

# مهندسی ویژگی (**Feature Engineering**)

در این مرحله جهت آماده سازی داده برای ورود به مرحله مدل سازی لازم است تمام ویژگی های ورودی به صورت بردار در قالب یک ویژگی تجمیع شوند. در واقع پس از این مرحله مجموعه داده دارای دو ستون خواهد بود: ستون اول تمام ویژگی های ورودی که به برداری از ویژگی ها تبدیل شده اند و ستون دوم خروجی یا برچسب هر رکورد. کد این مرحله به صورت زیر می باشد:

from pyspark.ml.feature import VectorAssembler

def feature\_engineering(input\_df,feature\_list,label\_name):

    assembler = VectorAssembler(inputCols=feature\_list,

                             outputCol='features')

    assembled\_df = assembler.transform(input\_df)

    output\_df=assembled\_df.select('features', label\_name)

    return output\_df

input\_features=list(set(prep\_df.columns) - set(['label','is\_match']))

assembled\_df = feature\_engineering(prep\_df,input\_features,'label')

خروجی مجموعه داده پس از انجام مهندسی ویژگی به صورت زیر است:

assembled\_df.show(10, truncate=False)

