

بسمه تعالی



نام درس:

کاوش دادگان انبوه (Big data)

نام پروژه:

**ساخت مدل پیش بینی بر روی یک مجموعه کلان داده (Big data) با استفاده از ابزار
یادگیری ماشینی برای کلان داده**

نام استاد درس:

دکتر الهام آخوندزاده

نام TA درس:

مهندس امیررضا نجفی

نام و شماره دانشجویی اعضای گروه:

سمیه حمیدی ۴۰۰۶۶۲۴۱۰۰۲

هاله خوش شانس ۴۰۰۶۶

جواد رفیعی فرد ۴۰۰۶۶۲۴۱۰۰۵

محمد رضا شاقوزی ۴۰۰۶۶۲۴۱۰۰۸

بسمه تعالی

گزارش پروژه درس داده کاوی

ویرایش: ۱۴۰۱/۱۱/۲۲ - تدوین: ۱۴۰۱/۱۱/۰۶ - تهیه کننده: سمیه حمیدی، هاله خوش شانس، جواد

رفیعی فرد، محمدرضا شاقوزی

فهرست

۳	۱ معرفی مجموعه داده و توصیف آن
۳	۱-۱ معرفی کلی مجموعه داده و نحوه شکل گیری آن
۴	۱-۲ معرفی ویژگی های ورودی
۵	۱-۳ ویژگی خروجی (هدف):
۵	۲ تحلیل اکتشافی داده (Exploratory data analysis)
۵	۲-۱ مجموعه داده در یک نگاه
۵	۳ فراخوانی کتابخانه های مورد نیاز و انجام تنظیمات pyspark
۶	۴ جمعیت چانک های مجموعه داده
۷	۵ مرحله پیش پردازش
۷	۵-۱ تبدیل به داده های عددی
۸	۵-۲ نرمال سازی
۹	۵-۳ تشخیص و حذف داده های پرت
۹	۶ داده های از دست رفته (Missing Values)
۹	۶-۱ بررسی داده های از دست رفته
۱۰	۶-۲ پر کردن داده های از دست رفته
۱۲	۷ مهندسی ویژگی (Feature Engineering)

۱ معرفی مجموعه داده و توصیف آن

۱-۱ معرفی کلی مجموعه داده و نحوه شکل گیری آن

در این تحقیق قصد داریم یک پروژه یادگیری ماشینی را بر روی یک مجموعه کلان داده (Big data) توسط روش‌های پردازش کلان داده انجام دهیم. این مجموعه داده که از سایت Machine Learning Repository استخراج شده، داده‌های مربوط به تعداد زیادی از بیماران در یک مرکز تحقیقات بیماران سرطانی در آلمان به نام Epidemiologisches Krebsregister NRW است و در آدرس زیر قابل دسترس است:

<https://archive.ics.uci.edu/ml/machine-learning-databases/00210/donation.zip>

موضوع از این قرار است که اطلاعات ۱۰۰,۰۰۰ بیمار بین سال‌های ۲۰۰۵ تا ۲۰۰۸ توسط کارمندان مختلف مرکز تحقیقات ثبت شده است. مشکل اینجاست که اطلاعات یک فرد ممکن است دوبار توسط کارمندان ثبت شده باشد و مشکل اساسی تر اینکه در این بین اشتباهات تایپی نیز وجود دارد.

بنابراین مسأله این مجموعه داده پیدا کردن جفت رکوردهایی از اطلاعات است که متعلق به یک شخص است.

پس از آن در سال ۲۰۰۸ موسسه آمار زیستی پزشکی، اپیدمیولوژی و انفورماتیک (IMBEI) و مرکز پزشکی دانشگاه یوهانس گوتنبرگ، ماینس، آلمان بر اساس ۶ معیار که در ادامه می‌آید تمام جفت رکوردهای مربوط به این ۱۰۰,۰۰۰ رکورد (بیمار) را مورد بررسی قرار دادند. معیارهای مذکور به شرح زیر هستند:

- ۱- برابری آوایی نام و نام خانوادگی + برابری تاریخ تولد
- ۲- برابری آوایی نام + برابری روز تولد (ماه و سال می‌توانند متفاوت باشند)
- ۳- برابری آوایی نام + برابری ماه تولد (روز و سال می‌توانند متفاوت باشند)
- ۴- برابری آوایی نام + برابری سال تولد (روز و ماه می‌توانند متفاوت باشند)
- ۵- برابری تاریخ تولد
- ۶- برابری آوایی نام خانوادگی + برابری جنسیت

برقراری هر کدام از شروط فوق منجر به انتخاب جفت رکورد شد. در نتیجه ۵,۷۴۹,۱۳۲ جفت رکورد از میان تمام جفت رکوردها انتخاب شد که از این بین در ۲۰,۹۳۱ جفت رکورد، شخص اول و شخص دوم هر دو یک نفر بودند که برچسب match به آنها تعلق گرفت یعنی این جفت رکورد مربوط به یک شخص است و در مابقی جفت رکوردها شخص اول و شخص دوم دو فرد متفاوت بودند.

این مجموعه داده به ۱۰ فایل با اندازه تقریباً برابر تقسیم شد که در هر فایل نسبت جفت رکوردهای match و non-match تقریباً برابر است.

لازم به ذکر است در مجموعه داده اولیه هر رکورد متعلق به یک بیمار بود که شامل ۱۰۰,۰۰۰ رکورد بود ولی پس از بررسی جفت رکوردها مجموعه داده جدیدی تشکیل شد که هر رکورد متعلق به رکورد مجموعه داده اولیه است که باهم مقایسه می‌شوند که تعداد آنها ۵,۷۴۹,۱۳۲ رکورد است.

از اینجا به بعد منظور از عبارت رکورد، هر سطر از مجموعه داده جدید (نهایی) است. هدف از مجموعه داده نهایی یافتن یک مدل یادگیری ماشینی بر اساس کلان داده برای پیش بینی وضعیت match بودن یا نبودن دو شخص بر اساس ویژگی های آنهاست که در ادامه به معرفی این ویژگی ها می پردازیم: مجموعه داده نهایی شامل ۱۲ ویژگی است که ۲ ویژگی اول آن شماره شناسایی دو شخص مورد مقایسه است که برای پیش بینی مورد استفاده قرار نمی گیرد ۹ ویژگی، ویژگی های ورودی مورد استفاده جهت پیش بینی و ۱ ویژگی، ویژگی خروجی است که match بودن یا نبودن دو شخص را مشخص می کند.

۱-۲ معرفی ویژگی های ورودی

۱- id_1 :

شماره شناسایی شخص اول که یک متغیر عددی و از نوع صحیح است.

۲- id_2 :

شماره شناسایی شخص دوم که یک متغیر عددی و از نوع صحیح است.

این دو ویژگی در ساخت مدل شرکت نمی کنند و در پیش پردازش از مجموعه داده حذف خواهند شد.

۳- cmp_fname_c1 :

میزان مشابهت جزء اول نام دو نفر که یک متغیر عددی پیوسته و از نوع اعشاری است.

۴- cmp_fname_c2 :

میزان مشابهت جزء دوم نام دو نفر که یک متغیر عددی پیوسته و از نوع اعشاری است.

۵- cmp_lname_c1 :

میزان مشابهت جزء اول نام خانوادگی دو نفر که یک متغیر عددی پیوسته و از نوع اعشاری است.

۶- cmp_lname_c2 :

میزان مشابهت جزء دوم نام خانوادگی دو نفر که یک متغیر عددی پیوسته و از نوع اعشاری است.

(Sariyar et al., ۲۰۱۲) در تحقیق خود اشاره کرده است که نام و نام خانوادگی دارای دو جزء است.

An example of a realistic and problematic record linkage task regarding personal data is given below with the following attributes: first name and last name (two **components** each), sex, date of birth (comprised of day, month and year) and postal code.

```
(( 'Peter', 'John', 'Branket', , 'm', '11', '10', '1971', '100098' )
( 'Peter', , 'Blanket', , 'm', '01', '10', '1971', '10098' ))
```

۷- cmp_sex :

تشابه جنسیت دو نفر که یک متغیر باینری است و مقدار آن ۰ یا ۱ است که ۰ به معنی عدم یکسانی جنسیت و ۱ به معنی یکسان بودن جنسیت دو نفر است.

۸- cmp_bd :

تشابه روز تولد دو نفر که یک متغیر باینری است و مقدار آن 0 یا 1 است که 0 به معنی عدم یکسانی روز تولد و 1 به معنی یکسان بودن روز تولد دو نفر است.

۹- cmp_bm :

تشابه ماه تولد دو نفر که یک متغیر باینری است و مقدار آن 0 یا 1 است که 0 به معنی عدم یکسانی ماه تولد و 1 به معنی یکسان بودن ماه تولد دو نفر است.

۱۰- cmp_by :

تشابه سال تولد دو نفر که یک متغیر باینری است و مقدار آن 0 یا 1 است که 0 به معنی عدم یکسانی سال تولد و 1 به معنی یکسان بودن سال تولد دو نفر است.

۱۱- cmp_plz :

تشابه کدپستی دو نفر که یک متغیر باینری است و مقدار آن 0 یا 1 است که 0 به معنی عدم یکسانی کدپستی و 1 به معنی یکسان بودن سال تولد دو نفر است.

۹ ویژگی فوق، در ساخت مدل شرکت می کنند.

۳-۱ ویژگی خروجی (هدف):

۱۲- is_match :

وضعیت تطابق دو نفر است که یک متغیر باینری است و مقدار آن True یا False است که True به معنی تطابق دو نفر و False به معنی عدم تطابق دو نفر است.

۲ تحلیل اکتشافی داده (Exploratory data analysis)

قرار هست خودم این قسمت را انجام بدهم 😊

۲-۱ مجموعه داده در یک نگاه

۳ 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()
```

۴ جمع چانک های مجموعه داده

مجموعه داده شامل ده فایل با مشخصات زیر می باشد:

 block_1.csv	2011/03/09 3:31 PM	CSV File	25,634 KB
 block_2.csv	2011/03/09 3:31 PM	CSV File	25,627 KB
 block_3.csv	2011/03/09 3:31 PM	CSV File	25,638 KB
 block_4.csv	2011/03/09 3:31 PM	CSV File	25,633 KB
 block_5.csv	2011/03/09 3:32 PM	CSV File	25,635 KB
 block_6.csv	2011/03/09 3:32 PM	CSV File	25,641 KB
 block_7.csv	2011/03/09 3:32 PM	CSV File	25,647 KB
 block_8.csv	2011/03/09 3:32 PM	CSV File	25,639 KB
 block_9.csv	2011/03/09 3:33 PM	CSV File	25,639 KB
 block_10.csv	2011/03/09 3:33 PM	CSV File	25,641 KB

این چانک ها را توسط کد زیر جمع 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()
```

۵ مرحله پیش پردازش

۵-۱ تبدیل به داده‌های عددی

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

```
df.printSchema()
```

```
root
|-- id_1: integer (nullable = true)
|-- id_2: integer (nullable = true)
|-- cmp_fname_c1: float (nullable = true)
|-- cmp_fname_c2: float (nullable = true)
|-- cmp_lname_c1: float (nullable = true)
|-- cmp_lname_c2: float (nullable = true)
|-- cmp_sex: integer (nullable = true)
|-- cmp_bd: integer (nullable = true)
|-- cmp_bm: integer (nullable = true)
|-- cmp_by: integer (nullable = true)
|-- cmp_plz: integer (nullable = true)
|-- is_match: boolean (nullable = true)
```

```
df.show(10)
```

id_1	id_2	cmp_fname_c1	cmp_fname_c2	cmp_lname_c1	cmp_lname_c2	cmp_sex	cmp_bd	cmp_bm	cmp_by	cmp_plz	is_match
3148	8326	1.0	null	1.0	null	1	1	1	1	1	true
14055	94934	1.0	null	1.0	null	1	1	1	1	1	true
33948	34740	1.0	null	1.0	null	1	1	1	1	1	true
946	71870	1.0	null	1.0	null	1	1	1	1	1	true
64880	71676	1.0	null	1.0	null	1	1	1	1	1	true
25739	45991	1.0	null	1.0	null	1	1	1	1	1	true
62415	93584	1.0	null	1.0	null	1	1	1	1	0	true
27995	31399	1.0	null	1.0	null	1	1	1	1	1	true
4909	12238	1.0	null	1.0	null	1	1	1	1	1	true
15161	16743	1.0	null	1.0	null	1	1	1	1	1	true

only showing top 10 rows

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()
```

```
root
 |-- id_1: integer (nullable = true)
 |-- id_2: integer (nullable = true)
 |-- cmp_fname_c1: float (nullable = true)
 |-- cmp_fname_c2: float (nullable = true)
 |-- cmp_lname_c1: float (nullable = true)
 |-- cmp_lname_c2: float (nullable = true)
 |-- cmp_sex: integer (nullable = true)
 |-- cmp_bd: integer (nullable = true)
 |-- cmp_bm: integer (nullable = true)
 |-- cmp_by: integer (nullable = true)
 |-- cmp_plz: integer (nullable = true)
 |-- label: integer (nullable = false)
```

```
numerical_df.show(10)
```

```
+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+
| id_1 | id_2 | cmp_fname_c1 | cmp_fname_c2 | cmp_lname_c1 | cmp_lname_c2 | cmp_sex | cmp_bd | cmp_bm | cmp_by | cmp_plz | label |
+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+
| 3148 | 8326 | 1.0 | null | 1.0 | null | 1 | 1 | 1 | 1 | 1 | 1 |
| 14055 | 94934 | 1.0 | null | 1.0 | null | 1 | 1 | 1 | 1 | 1 | 1 |
| 33948 | 34740 | 1.0 | null | 1.0 | null | 1 | 1 | 1 | 1 | 1 | 1 |
| 946 | 71870 | 1.0 | null | 1.0 | null | 1 | 1 | 1 | 1 | 1 | 1 |
| 64880 | 71676 | 1.0 | null | 1.0 | null | 1 | 1 | 1 | 1 | 1 | 1 |
| 25739 | 45991 | 1.0 | null | 1.0 | null | 1 | 1 | 1 | 1 | 1 | 1 |
| 62415 | 93584 | 1.0 | null | 1.0 | null | 1 | 1 | 1 | 1 | 0 | 1 |
| 27995 | 31399 | 1.0 | null | 1.0 | null | 1 | 1 | 1 | 1 | 1 | 1 |
| 4909 | 12238 | 1.0 | null | 1.0 | null | 1 | 1 | 1 | 1 | 1 | 1 |
| 15161 | 16743 | 1.0 | null | 1.0 | null | 1 | 1 | 1 | 1 | 1 | 1 |
+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+
only showing top 10 rows
```

۵-۲ نرمال سازی

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

```
numerical_df.describe().show()
```


summary	id_1	id_2	cmp_fname_c1	cmp_fname_c2	cr
cmp_sex	cmp_bd	cmp_bm	cmp_by	cmp_plz	
count	5749132	5749132	5748125	103698	
5749132	5748337	5748337	5748337	5736289	
mean	33324.48559643438	66587.43558331935	0.7129024717700259	0.900017672461932	0.31562
	0.955001381078048	0.22446526708507172	0.48885529849763504	0.2227485966810923	0.00552866147434
stddev	23659.859374488064	23620.487613269695	0.3887583583605727	0.27131760936040944	0.3342336
	0.4998758236779031	0.4160909629831756	0.07414914925420046	0.06022847555207964	
min	1	6	0.0	0.0	
0	0	0	0	0	
max	99980	100000	1.0	1.0	
1	1	1	1	1	

۵-۳ تشخیص و حذف داده های پرت

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

۶ داده های از دست رفته (Missing Values)

۶-۱ بررسی داده های 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)
```

cmp_fname_c1	cmp_fname_c2	cmp_lname_c1	cmp_lname_c2	cmp_sex	cmp_bd	cmp_bm	cmp_by	cmp_plz	label
1.0	null	1.0	null	1	1	1	1	1	1
1.0	null	1.0	null	1	1	1	1	1	1
1.0	null	1.0	null	1	1	1	1	1	1
1.0	null	1.0	null	1	1	1	1	1	1
1.0	null	1.0	null	1	1	1	1	1	1
1.0	null	1.0	null	1	1	1	1	1	1
1.0	null	1.0	null	1	1	1	1	0	1
1.0	null	1.0	null	1	1	1	1	1	1
1.0	null	1.0	null	1	1	1	1	1	1
1.0	null	1.0	null	1	1	1	1	1	1

توسط کد زیر وضعیت داده های 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:

id_1	id_2	cmp_fname_c1	cmp_fname_c2	cmp_lname_c1	cmp_lname_c2	cmp_sex	cmp_bd	cmp_bm	cmp_by	cmp_plz	label
0	0	1007	5645434	0	5746668	0	795	795	795	12843	0

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 داده های از دست رفته داشته باشد وجود ندارد:
 تعداد کل رکوردها:

```
no_id_numerical_df.count()
```

5749132

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

```
no_id_numerical_df.na.drop(how='all').count()
```

5749132

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

```
miss_df.na.drop(how='any', thresh=2).count()
```

5749132

۶-۲ پر کردن داده های از دست رفته

همانگونه که در قسمت قبل ذکر شد مجموعه داده دارای داده های از دست رفته زیادی مخصوصاً در ویژگی های cmp_fname_c2 و cmp_lname_c2 است:

```
no_id_numerical_df.show(10)
```

cmp_fname_c1	cmp_fname_c2	cmp_lname_c1	cmp_lname_c2	cmp_sex	cmp_bd	cmp_bm	cmp_by	cmp_plz	label
1.0	null	1.0	null	1	1	1	1	1	1
1.0	null	1.0	null	1	1	1	1	1	1
1.0	null	1.0	null	1	1	1	1	1	1
1.0	null	1.0	null	1	1	1	1	1	1
1.0	null	1.0	null	1	1	1	1	1	1
1.0	null	1.0	null	1	1	1	1	1	1
1.0	null	1.0	null	1	1	1	1	0	1
1.0	null	1.0	null	1	1	1	1	1	1
1.0	null	1.0	null	1	1	1	1	1	1
1.0	null	1.0	null	1	1	1	1	1	1

نمونه داده قبل از پر کردن داده های از دست رفته

با استفاده از روش Imputer داده های 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:

cmp_fname_c1_imp	cmp_fname_c2_imp	cmp_lname_c1_imp	cmp_lname_c2_imp	cmp_sex_imp	cmp_bd_imp	cmp_bm_imp	cmp_by_imp	cmp_plz_imp	label
1.0	0.9000177	1.0	0.31841284	1	1	1	1	1	1
1.0	0.9000177	1.0	0.31841284	1	1	1	1	1	1
1.0	0.9000177	1.0	0.31841284	1	1	1	1	1	1
1.0	0.9000177	1.0	0.31841284	1	1	1	1	1	1
1.0	0.9000177	1.0	0.31841284	1	1	1	1	1	1
1.0	0.9000177	1.0	0.31841284	1	1	1	1	1	1
1.0	0.9000177	1.0	0.31841284	1	1	1	1	1	1
1.0	0.9000177	1.0	0.31841284	1	1	1	1	1	1
1.0	0.9000177	1.0	0.31841284	1	1	1	1	1	1
1.0	0.9000177	1.0	0.31841284	1	1	1	1	1	1
1.0	0.9000177	1.0	0.31841284	1	1	1	1	1	1
1.0	0.9000177	1.0	0.31841284	1	1	1	1	1	1
1.0	0.9000177	1.0	0.31841284	1	1	1	1	1	1
1.0	0.9000177	1.0	0.31841284	1	1	1	1	1	1

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:

cmp_fname_c1_imp	cmp_fname_c2_imp	cmp_lname_c1_imp	cmp_lname_c2_imp	cmp_sex_imp	cmp_bd_imp	cmp_bm_imp	cmp_by_imp	cmp_plz_imp	label
0	0	0	0	0	0	0	0	0	0

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

۷ 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(prepare_df.columns) - set(['label', 'is_match']))
assembled_df = feature_engineering(prepare_df,input_features, 'label')
```

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

```
assembled_df.show(10, truncate=False)
```

features	label
[1.0,1.0,1.0,0.3184128403663635,1.0,1.0,1.0,0.9000176787376404,1.0]	1
[1.0,1.0,1.0,0.3184128403663635,1.0,1.0,1.0,0.9000176787376404,1.0]	1
[1.0,1.0,1.0,0.3184128403663635,1.0,1.0,1.0,0.9000176787376404,1.0]	1
[1.0,1.0,1.0,0.3184128403663635,1.0,1.0,1.0,0.9000176787376404,1.0]	1
[1.0,1.0,1.0,0.3184128403663635,1.0,1.0,1.0,0.9000176787376404,1.0]	1
[1.0,1.0,1.0,0.3184128403663635,1.0,1.0,1.0,0.9000176787376404,1.0]	1
[1.0,1.0,1.0,0.3184128403663635,1.0,1.0,1.0,0.9000176787376404,1.0]	1
[1.0,1.0,1.0,0.3184128403663635,1.0,1.0,1.0,0.9000176787376404,0.0]	1
[1.0,1.0,1.0,0.3184128403663635,1.0,1.0,1.0,0.9000176787376404,1.0]	1
[1.0,1.0,1.0,0.3184128403663635,1.0,1.0,1.0,0.9000176787376404,1.0]	1
[1.0,1.0,1.0,0.3184128403663635,1.0,1.0,1.0,0.9000176787376404,1.0]	1