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 40066

جواد رفیعی فرد 40066241005

محمدرضا شاقوزی 40066241008

بسمه تعالی

گزارش پروژه درس داده کاوی

ویرایش: 22/11/1401 – تدوین: 06/11/1401 – تهیه‏کننده: سمیه حمیدی، هاله خوش شانس، جواد رفیعی فرد، محمدرضا شاقوزی

**فهرست**

[1 معرفی مجموعه داده و توصیف آن 3](#_Toc127009604)

[1-1 معرفی کلی مجموعه داده و نحوه شکل گیری آن 3](#_Toc127009605)

[1-2 معرفی ویژگی های ورودی 4](#_Toc127009606)

[1-3 ویژگی خروجی (هدف): 5](#_Toc127009607)

[2 تحلیل اکتشافی داده (**Exploratory data analysis**) 5](#_Toc127009608)

[2-1 مجموعه داده در یک نگاه 5](#_Toc127009609)

[**3** فراخوانی کتابخانه های مورد نیاز و انجام تنظیمات **pyspark** 5](#_Toc127009610)

[4 تجمیع چانک های مجموعه داده 6](#_Toc127009611)

[5 مرحله پیش پردازش 7](#_Toc127009612)

[5-1 تبدیل به داده‌های عددی 7](#_Toc127009613)

[5-2 نرمال سازی 8](#_Toc127009614)

[5-3 تشخیص و حذف داده های پرت 9](#_Toc127009615)

[6 داده‏های از دست رفته (**Missing Values**) 9](#_Toc127009616)

[6-1 بررسی داده‏های از دست رفته 9](#_Toc127009617)

[6-2 پر کردن داده‏های از دست رفته 10](#_Toc127009618)

[7 مهندسی ویژگی (**Feature Engineering**) 12](#_Toc127009619)

# معرفی مجموعه داده و توصیف آن

## معرفی کلی مجموعه داده و نحوه شکل گیری آن

در این تحقیق قصد داریم یک پروژه یادگیری ماشینی را بر روی یک مجموعه کلان‏داده (Big data) توسط روش‏های پردازش کلان داده انجام دهیم. این مجموعه داده که از سایت Machine Learning Repository استخراج شده، داده های 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 100،000 بیمار بین سال های 2005 تا 2008 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 2008 موسسه آمار زیستی پزشکی، اپیدمیولوژی و انفورماتیک (IMBEI) و مرکز پزشکی دانشگاه یوهانس گوتنبرگ، ماینتس، آلمان بر اساس 6 معیار که در ادامه می آید تمام جفت رکوردهای مربوط به این 100،000 رکورد (بیمار) را مورد بررسی قرار دادند.

معیارهای مذکور به شرح زیر هستند:

1. برابری آوایی نام و نام خانوادگی + برابری تاریخ تولد
2. برابری آوایی نام + برابری روز تولد (ماه و سال می توانند متفاوت باشند)
3. برابری آوایی نام + برابری ماه تولد (روز و سال می توانند متفاوت باشند)
4. برابری آوایی نام + برابری سال تولد (روز و ماه می توانند متفاوت باشند)
5. برابری تاریخ تولد
6. برابری آوایی نام خانوادگی + برابری جنسیت

برقراری هر کدام از شروط فوق منجر به انتخاب جفت رکورد شد. در نتیجه 5،749،132 جفت رکورد از میان تمام جفت رکوردها انتخاب شد که از این بین در 20،931 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 10 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 100،000 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 می شوند که تعداد آنها 5،749،132 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 می‏پردازیم:

مجموعه داده نهایی شامل 12 ویژگی است که 2 ویژگی اول آن شماره شناسایی دو شخص مورد مقایسه است که برای پیش‏بینی مورد استفاده قرار نمی‏گیرد 9 ویژگی، ویژگی های ورودی مورد استفاده جهت پیش بینی و 1 ویژگی، ویژگی خروجی است که match بودن یا نبودن دو شخص را مشخص می کند.

## معرفی ویژگی های ورودی

1. id\_1 :

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

1. 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.

1. cmp\_fname\_c1:

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

1. cmp\_fname\_c2:

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

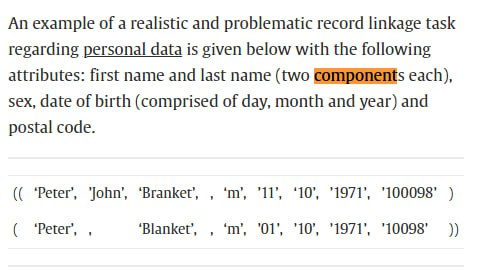
1. cmp\_lname\_c1:

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

1. cmp\_lname\_c2:

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

(Sariyar *et al.*, 2012) در تحقیق خود اشاره کرده است که نام و نام خانوادگی دارای دو جزء است.



1. cmp\_sex:

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

1. 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.

1. 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.

1. 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.

1. 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.

9 ویژگی فوق، در ساخت مدل شرکت می کنند.

## ویژگی خروجی (هدف):

1. 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.

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

قرار هست خودم این قسمت را انجام بدهم 😊

## مجموعه داده در یک نگاه

# 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\

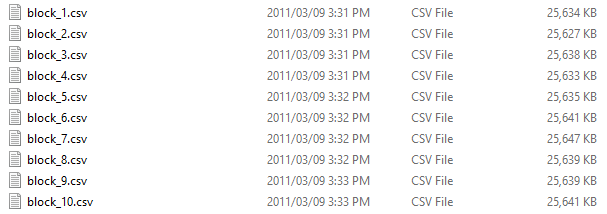
    .appName('myApp')\

        .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)

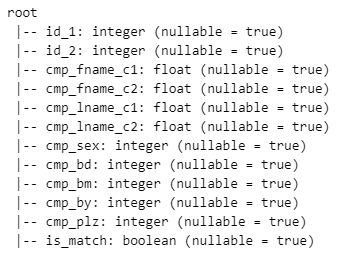
df=load\_record\_linkage\_data()

# مرحله پیش پردازش

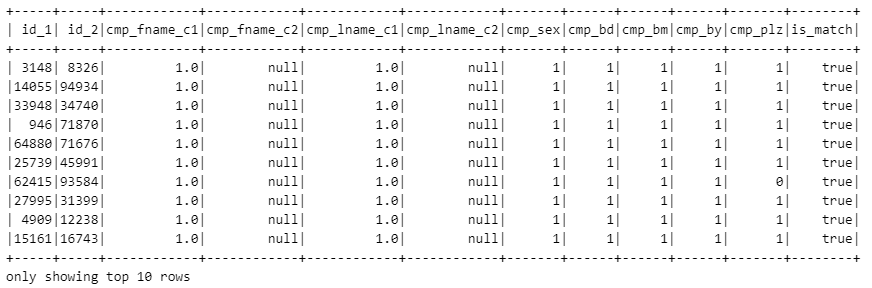
## تبدیل به داده‌های عددی

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)

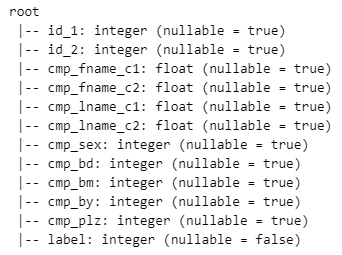
                                  )

    return temp

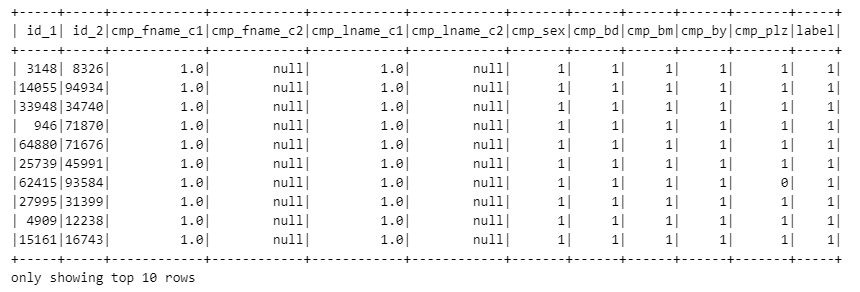
numerical\_df = convert\_label\_binary(df).drop('is\_match')

که نتیجه آن به صورت زیر می باشد:

numerical\_df.printSchema()



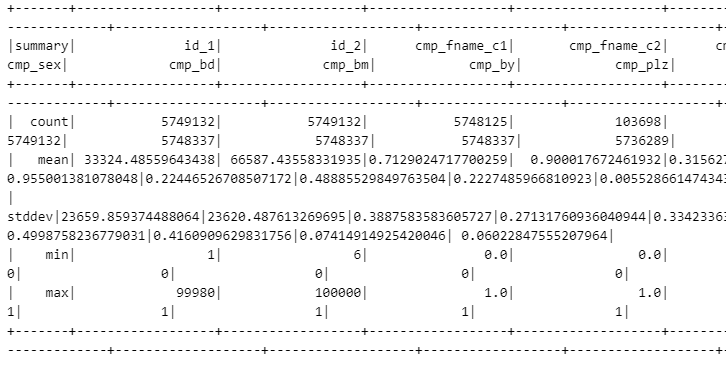
numerical\_df.show(10)



## 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**)

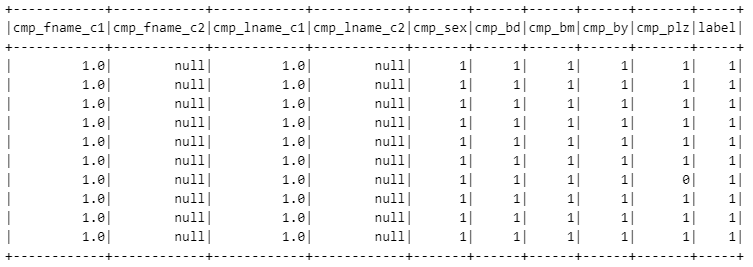
## بررسی داده‏های از دست رفته

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:

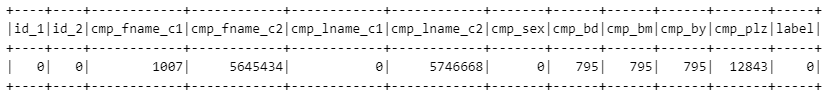
no\_id\_numerical\_df.show(10)



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:

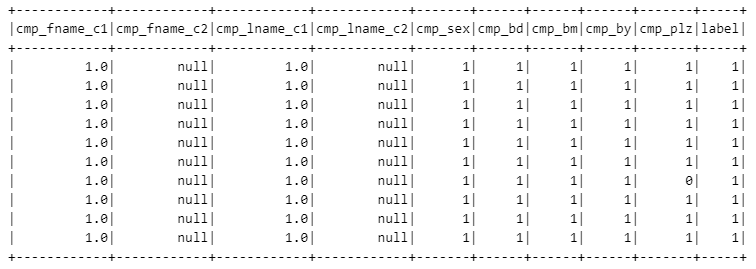
miss\_df.na.drop(how='any',thresh=2).count()



## 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'])

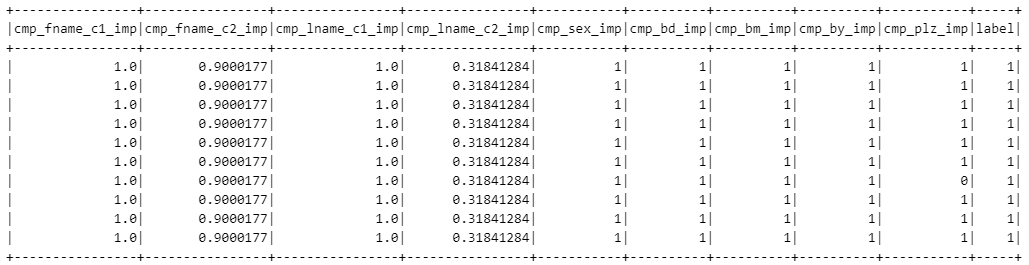
    return output\_df

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:

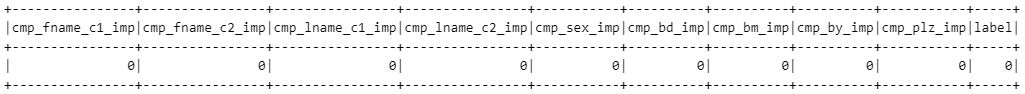


نمونه داده پس از پر کردن داده‏های از دست رفته

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)

