بسم الله الرحمن الرحیم

پروژه سوم درس یادگیری ماشین توزیع شده

دکتر دوستی

مهدی وجهی - ۸۱۰۱۰۱۵۵۸

فهرست

سوال ۱ 3
بخش ۱ 3
بخش دوم 4
بخش ۳ 6
سوال ۲ 8
بخش ۱ 8
بخش ۲ 8
بخش ۳ 9
بخش ۴ 10
سوال ۳ 11
بخش ۱ 11
بخش ۲ 11
بخش ۳ 12
بخش ۴ 14

سوال ۱

در ابتدا هدر را جدا می کنیم و تبدیل به یک دیکشنری می کنیم که ایندکس را برگرداند سپس هدر را حذف می کنیم و تمامی ردیف ها را طبق ',' جدا می کنیم و در نهایت تنها مواردی که نام دارند را نگه می داریم:

```
raw_data = sc.textFile("video_game_sales.csv")
header_raw = raw_data.first()
header = {column: index for index, column in
enumerate(header_raw.split(','))}
raw_data = raw_data.filter(lambda line: line != header_raw) \
   .map(lambda line: line.split(',')) \
   .filter(lambda x: bool(x[header['title']]))
```

بخش ۱

در این بخش برای استخراج ۱۰ بازی پرفروش اول باید داده های خالی را پر کنیم. ما نیاز به نام ها داریم (که پر هستند) و نیاز به فروش کل داریم (که می تواند خالی یا نامعتبر باشد). برای پر کردن فروش کل میتوانیم اگر فروش در تمام مناطق موجود باشد از آن استفاده کنیم. برای این کار ابتدا نام بازی و فروش کل و فروش به تفکیک مناطق را جدا می کنیم.

```
.map(lambda x: (x[header['title']], *x[7:12]))
```

سپس در تابعی اگر فروش کل موجود بود از آن استفاده می کنیم و اگر نبود از سعی می کنیم از ریز فروش آن را استخراج کنیم. اگر هم که امکانش نبود خالی بر می گردانیم.

```
def calc_total_sales(x):
   try:
       return (x[0], float(x[1]))
   except:
       pass

   try:
       return (x[0], sum(map(float, x[2:])))
   except:
       pass

   return None
```

البته این کار به هیچ عنوان بهینه نیست زیرا باید در هر گام پردازه پایتونی ایجاد شود داده ها سریال شوند و برای آن ارسال شوند و همچنین منطق تابع با try/catch هست که سربار زیادی دارد در آخر نیز مجدد لازم است داده ها سریال و ارسال شوند. این کار به شدت کند است.

در این مرحله تابع را صدا می کنیم و نتایج خالی را دور میریزیم.

```
   .map(calc_total_sales) \
   .filter(lambda x: x != None)
```

در نهایت چون اسم بعضی از بازی ها تکراری است و احتمالا فروش در سکو های مختلف است آن ها را کاهش می دهیم و جمع می زنیم. در مرحله آخر هم نتایج را مرتب می کنیم.

```
    .reduceByKey(lambda a, b: a+b) \
    .sortBy(lambda x: x[1], ascending=False)
```

در آخر خروجی زیر را دریافت کردیم.

```
[('"Warhammer 40', 124.700),
 ('Grand Theft Auto V', 64.290),
 ("You Don't Know Jack", 47.600),
 ('Call of Duty: Black Ops', 30.990),
 ('Call of Duty: Modern Warfare 3', 30.710),
 ('Call of Duty: Black Ops II', 29.590),
 ('Call of Duty: Ghosts', 28.800),
 ('Call of Duty: Black Ops 3', 26.720),
 ('Aggressive Inline', 26.610),
 ('The Lord of the Rings: The Return of the King', 26.460)]
```

بخش دوم

این قسمت بسیار مشابه قسمت قبل است و مجدد داده های فروش را انتخاب می کنیم و به جای نام بازی ژانر را انتخاب می کنیم.

```
raw_data.map(lambda x: (x[header['genre']], *x[7:12]))
```

در اینجا لازم است پردازشی انجام دهیم. اگر تمام داده های فروش کامل بود که قابل استفاده است و اگر نبود باید بررسی کرد که اگر تنها یکی از آنها خالی است می توان از سایر موارد آن را استخراج کرد و اگر بیشتر بود باید آن را دور ریخت. ابتدا یک تابع تعریف می کنیم که ژانر را جدا می کند و تابع درونی را صدا می زند و اگر موفق بود کل داده را بر می گرداند.

```
def calc_sales_regen(x):
    sales = fix_sales_list(x[1:])
    if sales is None:
        return None
    else:
        return (x[0], sales)
```

سپس تعداد خانه های نامعتبر را پیدا می کنیم و نسبت به آن تصمیم می گیریم.

```
def fix_sales_list(x):
    invalid_indices = []

    for i, item in enumerate(x):
        try:
            float(item)
        except:
            invalid_indices.append(i)
```

```
    if len(invalid_indices) > 1:
        return None
    elif len(invalid_indices) == 0:
        return list(map(float, x))
```

اگر نیاز به بازسازی داشت (یک ستون ناموجود بود) این کار را انجام می دهیم.

```
    try:
        if missing_idx == 0:
            current_sum = 0
            for i in range(1, len(x)):
                current_sum += float(x[i])
            calculated_value = current_sum

        else:
            total_sum = float(x[0])
            current_sum = 0
            for i in range(1, len(x)):
                if i == missing_idx:
                    continue
                current_sum += float(x[i])

            calculated_value = total_sum - current_sum

        x[missing_idx] = calculated_value
        return list(map(float, x))

    except:
        return None
```

سپس این تابع را روی داده های اعمال می کنیم و موارد نامطلوب را دور می ریزیم.

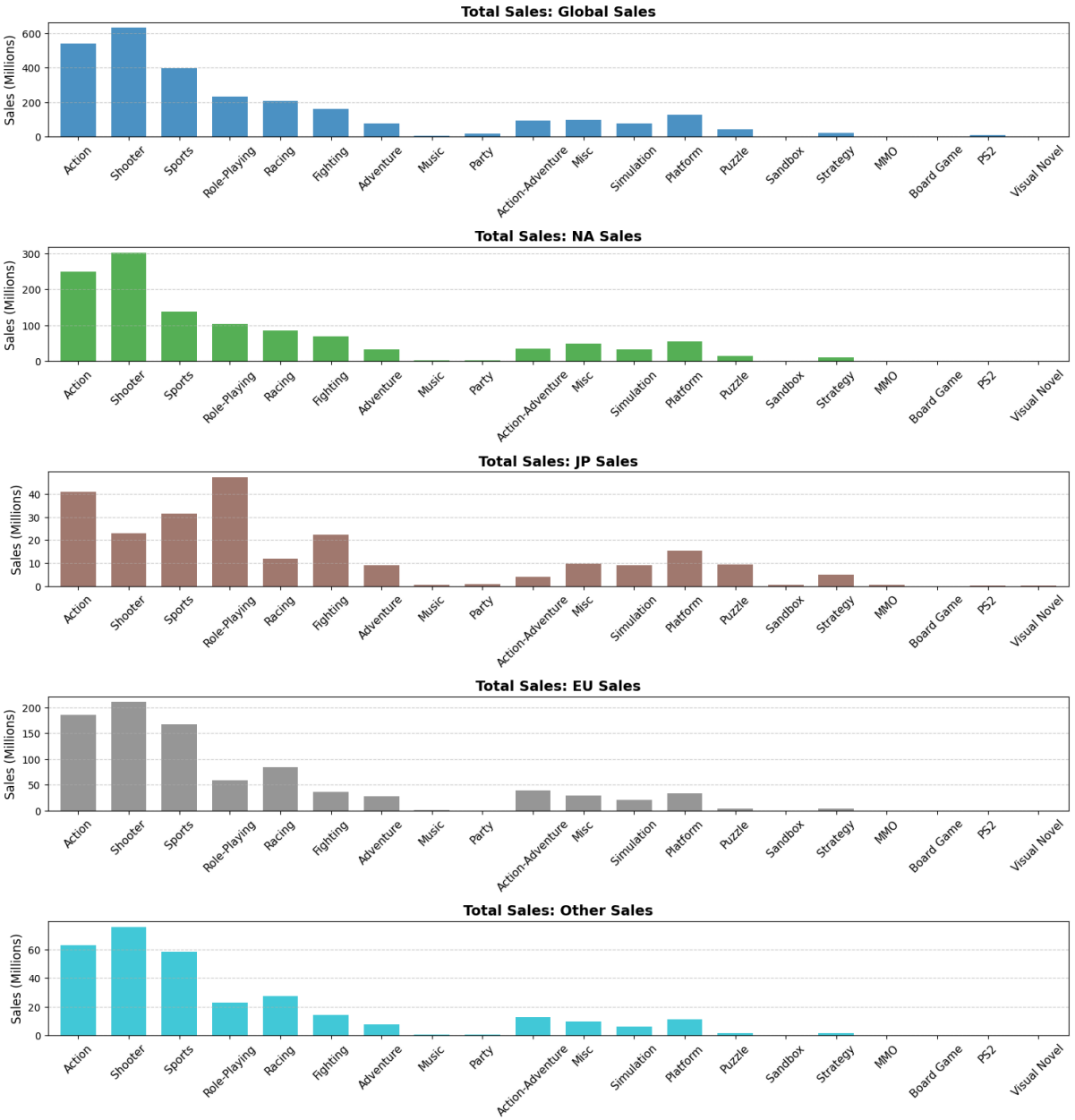
```
    .map(calc_sales_regen) \
    .filter(lambda x: x != None)
```

در آخر هم فروش ژانر ها را تجمیع می کنیم.

```
.reduceByKey(lambda list1, list2: tuple(a + b for a, b in zip(list1,
list2)))
```

در آخر هم نموداری رسم می کنیم به شکل زیر:

Sales Distribution per Genre by Region



بخش ۳

ابتدا تنها مواردی که امتیاز دارند را فیلتر می کنیم.

```
def have_score(x):
    try:
        float(x[header['critic_score']])
        return True
    except:
        return False
```

```
raw_data.filter(have_score)
```

سپس مانند بخش های قبل میزان فروش را بازسازی می کنیم و اگر نشد صفر قرار می دهیم. (حذف نمی کنیم زیرا میانگین امتیاز و مجموع فروش نسبتا مستقل هستند و می توان جداگانه نگاه و تحلیل کرد.) همچنین برای شمارش تعداد برای میانگین گیری یک ستون ۱ نیز به آن اضافه می کنیم.

```
def calc_total_sales(x):
   try:
       return float(x[header['total_sales']])
   except:
       pass

   try:
       return sum(map(float, x[header['total_sales']+1:
header['other_sales']+1]))
   except:
       pass

   return 0

.map(lambda x: (x[header['developer']], (calc_total_sales(x),
float(x[header['critic_score']]), 1)))
```

حال داده های را تجمیع می کنیم و میانگین امتیاز را حساب می کنیم.

```
   .reduceByKey(lambda list1, list2: tuple(a + b for a, b in zip(list1,
list2))) \
   .mapValues(lambda x: (x[0], x[1]/x[2]))
```

در آخر هم شرط حداقل میزان فروش را اعمال می کنیم و نتایج را مرتب می کنیم.

```
   .filter(lambda x: x[1][0] >= MIN_SALES) \
   .sortBy(lambda x: x[1][1], ascending=False)
```

نتیجه به شکل زیر است:

```
[('Rockstar Games', (13.940, 9.800)),
 ('Naughty Dog', (20.140, 9.083)),
 ('Rockstar North', (99.570, 9.061)),
 ('Rocksteady Studios', (22.590, 9.025)),
 ('Bungie', (13.380, 8.900)),
 ('Turn 10 Studios', (12.120, 8.900)),
 ('Rockstar San Diego', (22.270, 8.722)),
 ('Dice', (17.320, 8.633)),
 ('Harmonix Music Systems', (29.330, 8.544)),
 ('EA Digital Illusions CE', (12.250, 8.480))]
```

سوال ۲

بخش ۱

بعد از خواندن فایل آن را به صورت خیلی ساده و با استفاده از فاصله ها توکنایز می کنیم.

```
split_data = raw_data.select(f.split(f.col('value'), ' ').alias('value'))
```

سپس مدل را آموزش می دهیم.

```
vectorSize=200
windowSize=5
minCount=3
numPartitions=11
maxIter=20
w2v = Word2Vec(
   seed=42,
   inputCol="value",
   outputCol="model",

   vectorSize=vectorSize,
   windowSize=windowSize,
   minCount=minCount,
   numPartitions=numPartitions,
   maxIter=maxIter
   )
```

اندازه وکتور همان فضای معنایی ما است که در مقاله ۳۰۰ است اما در اینجا چون هم داده کم است به نظرم عدد ۲۰۰ مناسب است. اندازه پنجره هم همان ۵ پیش فرض مناسب است و حداقل تکرار هم ۳ مناسب است و پارتیشن هم طبق تعداد هسته های cpu تعیین کردم و ۲۰ ایپاک داده را پیمایش می کنیم.
سپس مدل را آموزش و ذخیره می کنیم.

```
model = w2v.fit(split_data)
model.save(f'word2vec_VS{vectorSize}_WS{windowSize}_MC{minCount}_MI{maxIter}
_{datetime.datetime.now().strftime("D%Y_%m_%d_T%H_%M")}')
```

بخش ۲

برای این بخش کافیست یک فیلتر بزن بین موارد موجود در لیست مورد نظر سوال.

```
selected_words = ['iran', 'tehran', 'learning', 'science']
model.getVectors().filter(f.col('word').isin(selected_words)).show(truncate=
100)
```

خروجی:

```
+--------+----------------------------------------------------------------------------------------------------+
|    word|                                                                                              vector|
```

```
+--------+----------------------------------------------------------------------------------------------------+
|    iran|[0.2611589729785919,-0.26786503195762634,-0.21497038006782532,0.1693626195192337,-0.1699647605419...|
| science|[0.1457558423280716,0.18273581564426422,0.15451723337173462,-0.15501490235328674,0.11617875099182...|
|  tehran|[0.5012955665588379,-0.03322823718190193,-0.0753512904047966,0.5106785893440247,-0.22006152570247...|
|learning|[0.021161837503314018,0.44284340739250183,0.08777477592229843,-0.3333715796470642,0.2834542393684...|
+--------+----------------------------------------------------------------------------------------------------+
```

بخش ۳

مجدد به ازای تمام آن کلمات متد شباهت را صدا می زنیم (که طبق مستندات شباهت کسینوسی را اندازه می گیرد) سپس با وکتورشون جوین می کنیم.

```
for i in selected_words:
    print(i)
    model.findSynonyms(i, 5).join(model.getVectors(), 'word').show(5, 100)
```

خروجی:

```
iran
+----------+-----------------+----------------------------------------------------------------------------------------------------+
|      word|       similarity|                                                                                              vector|
+----------+-----------------+----------------------------------------------------------------------------------------------------+
|   pakistan|0.5902189016342163|[0.01546500250697136,-0.18069183826446533,-0.08781564235687256,0.19112782180309296,-0.08888671547...|
|    iranian| 0.478732705116272|[0.16548150777816772,0.03301917761564255,-0.16876323521137238,-0.3054888844490051,0.1121090352535...|
|      iraq|0.5148853063583374|[0.061682265251874924,-0.16518108546733856,-0.07135281711816788,0.01638515666127205,0.02287589013...|
|     syria|0.5015237927436829|[0.13604576885700226,-0.6059649586677551,-0.17501457035541534,-0.028692834079265594,0.24407251179...|
|afghanistan|0.5801478624343872|[0.20171602070331573,-0.24365027248859406,0.09517945349216461,0.18121525645256042,-0.054910574108...|
+----------+-----------------+----------------------------------------------------------------------------------------------------+

tehran
+---------+-----------------+----------------------------------------------------------------------------------------------------+
|     word|       similarity|                                                                                              vector|
+---------+-----------------+----------------------------------------------------------------------------------------------------+
|    jeddah|0.47724097967147827|[0.47635170817375183,0.027842691168189005,-0.24967725574970245,-0.2532123029232025,-0.372087478637...|
|     accra|0.44458678364753723|[0.39662715792655945,-0.3197892904281616,-0.2616196870803833,0.45456260442733765,-0.0626651197671...|
|  istanbul|0.44995713233947754|[0.12118062376976013,0.15797819197177887,-0.5601809024810791,0.5034356713294983,0.234818816184997...|
|dusseldorf| 0.4523930847644806|[0.15115147829055786,0.09853902459144592,0.038076236844062805,0.22718612849712372,-0.799529492855...|
|  chisinau|0.44119977951049805|[-0.09246578812599182,0.15801504254341125,-0.1263652890920639,0.06209447234869003,0.0816857218742...|
+---------+-----------------+----------------------------------------------------------------------------------------------------+

learning
+----------+-----------------+----------------------------------------------------------------------------------------------------+
|      word|       similarity|                                                                                              vector|
+----------+-----------------+----------------------------------------------------------------------------------------------------+
| vocational|0.43295395374298096|[0.20496653020381927,0.5086355805397034,-0.17237940430641174,-0.666351318359375,0.341406106948852...|
| montessori| 0.4428502917289734|[-0.07003387808799744,0.40951278805732727,-0.04785775765776634,-0.6152541041374207,0.095848195254...|
|educational| 0.4987924098968506|[-0.04645820707082748 4,0.3235059082508087,-0.172844797372818,-0.2934342324733734,0.35010048747062...|
|  education| 0.4539998471736908|[-0.15238501131534576,0.2325889766216278,0.01778140664100647,-0.5715274810791016,0.08364238590002...|
|    skills|0.47505220770835876|[-0.1114797443151474,0.3967192471027374,0.022748062387108803,-0.09434209018945694,0.2121769785881...|
+----------+-----------------+----------------------------------------------------------------------------------------------------+

science
+------------+-----------------+----------------------------------------------------------------------------------------------------+
|        word|       similarity|                                                                                              vector|
+------------+-----------------+----------------------------------------------------------------------------------------------------+
|anthropology|0.5277754664421082|[0.4610588550567627,0.2128603160381317,0.3301967978477478,-0.15136365592479706,-0.325629711151123...|
| mathematics|0.5373182892799377|[0.18220581114292145,-0.060965120792388916,-0.14754094183444977,-0.12785378098487854,-0.258761227...|
|    sciences| 0.6084183454513555|[-0.1454285830259323,0.0399218387901783,-0.20353825390338898,-0.43593719601631165,-0.350875884294...|
|  humanities|0.5414108633995056|[0.0665922611951828,0.13013219833374023,-0.4057573974132538,-0.5677540898323059,-0.20385284721851...|
|     fiction|0.6225622892379761|[0.39104875922203064,0.28393420577049255,0.31039172410964966,-0.002493906067684293,0.193484053015...|
+------------+-----------------+----------------------------------------------------------------------------------------------------+
```

بخش ۴

برای این بخش یک تابع نوشتیم. در این تابع ابتدا بردار کل این کلمات را دریافت می کند و در قالب یک دیکشنری ذخیره می کند. سپس از یک بردار صفر شروع می کند موارد مثبت را جمع می زند و منفی را تفریق می کند. سپس نتیجه را در متد شباهت می گذاریم و نتیجه را نمایش می دهیم.

```
def vac_calc(model, pos_words, neg_words):
   all_vec = model.getVectors().filter(f.col('word').isin(pos_words +
neg_words)).collect()
   all_vec = {i.word: i.vector for i in all_vec}

   result = np.zeros(model.getVectorSize())

   for i in pos_words:
       result += all_vec[i]

   for i in neg_words:
       result -= all_vec[i]

   return result

pos_words = ['king', 'woman']
neg_words = ['man']

model.findSynonyms(vac_calc(model, pos_words, neg_words), 5).show()
```

خروجی:

```
+-------+-------------------+
|   word|         similarity|
+-------+-------------------+
|   king| 0.7492444515228271|
|  queen| 0.4900623857975006|
|amalric|  0.485809862613678|
|  elara| 0.4716344475746155|
|  woman|0.47139328718185425|
+-------+-------------------+
```

همانطور که می بینید ملکه در جایگاه دوم است اما اگر کلمات نزدیک به شاه را بیابیم داریم:

```
+-------+------------------+
|   word|        similarity|
+-------+------------------+
| afonso|0.5781086683273315|
|     iv|0.5645919442176819|
|    iii|0.5424816608428955|
|mongkut|0.5390035510063171|
| youmiu|0.5387355089187622|
+-------+------------------+
```

که اصلا شامل ملکه نیست.

سوال ۳

بخش ۱

برای بارگزاری فایل در hdfs کافیست از دستور put استفاده کنیم.

```
hdfs dfs -put Titanic_Dataset.csv /vajhi/
```

نتیجه:

```
Found 1 items
-rw-r--r--   1 vajhi dmls   52460705 2025-12-25 15:44 /vajhi/wiki_corpus
**********
**********
Found 2 items
-rw-r--r--   1 vajhi dmls      61194 2025-12-26 14:21 /vajhi/Titanic_Dataset.csv
-rw-r--r--   1 vajhi dmls   52460705 2025-12-25 15:44 /vajhi/wiki_corpus
```

بخش ۲

با این دستور به کلاستر وصل می شویم:

```
spark = SparkSession.builder \
    .appName("Titanic") \
    .master("spark://raspberrypi-dml0:7077") \
    .config("spark.cores.max", "16") \
    .config("spark.executor.cores", "4") \
    .config("spark.executor.memory", "5500m") \
    .config("spark.driver.memory", "4g") \
    .config("spark.memory.fraction", "0.8") \
    .getOrCreate()
```

در ابتدا نام نشست و برنامه در مرحله بعد آدرس گره اصلی و در ادامه هر ۱۶ هسته کلاستر (در هر گره ۴ تا) را اختصاص می دهیم و در هر گره ۵.۵ گیگ رم اختصاص می دهیم و به درایور ۴ گیگ. همچنین ترکیب ممووری را به این صورت قرار می دهیم که ۲۰ درصد برای اشیا جاوا و ۸۰ درصد برای جداول و دیتا فریم ها.
در این مرحله فایل را می خوانیم:

```
raw_data = spark.read \
    .option("header", True) \
    .option("inferSchema", True) \
    .csv('hdfs://raspberrypi-dml0:9000/vajhi/Titanic_Dataset.csv')
```

با این روش می گوییم که ردیف اول هدر است و همچنین سعی کن نوع هر ستون را استخراج کنی و داده را به آن تبدیل کنی.

در نهایت برای پاسخ به سوال هم کافیست که جدول را طبق خواسته سوال گروه بندی کنیم و مورد مد نظر را میانگین بگیریم.

```
raw_data.groupBy('Sex').avg('Survived').show()
raw_data.groupBy('Pclass').avg('Survived').show()
raw_data.groupBy('Survived').avg('Age').show()
```

خروجی:

```
+------+-------------------+
|   Sex|      avg(Survived)|
+------+-------------------+
|female| 0.7420382165605095|
|  male|0.18890814558058924|
+------+-------------------+

+------+-------------------+
|Pclass|      avg(Survived)|
+------+-------------------+
|     1| 0.6296296296296297|
|     3|0.24236252545824846|
|     2|0.47282608695652173|
+------+-------------------+

+--------+-----------------+
|Survived|         avg(Age)|
+--------+-----------------+
|       1|28.343689655172415|
|       0| 30.62617924528302|
+--------+-----------------+
```

بخش ۳

در این قسمت ابتدا لازم است درکی از ساختار کلی داده و خانه های خالی پیدا کنیم. این دو دستور به ترتیب دقیقا همین کار را می کنند.

```
raw_data.select([f.count(f.when(f.col(c).isNull(), c)).alias(c) for c in
raw_data.columns]).show()
raw_data.summary().show()
```

خروجی:

```
+-----------+--------+------+----+---+---+-----+-----+------+----+-----+--------+
|PassengerId|Survived|Pclass|Name|Sex|Age|SibSp|Parch|Ticket|Fare|Cabin|Embarked|
+-----------+--------+------+----+---+---+-----+-----+------+----+-----+--------+
|          0|       0|     0|   0|  0|177|    0|    0|     0|   0|  687|       2|
+-----------+--------+------+----+---+---+-----+-----+------+----+-----+--------+

+-------+-----------+-------------------+-------------------+--------------------+------+-------+------------------+-------------------+-------------------+------------------+------------------+-----+--------+
|summary|PassengerId|           Survived|             Pclass|                Name|   Sex|    Age|             SibSp|              Parch|             Ticket|              Fare|             Cabin|Embarked|
+-------+-----------+-------------------+-------------------+--------------------+------+-------+------------------+-------------------+-------------------+------------------+------------------+-----+--------+
|  count|        891|                891|                891|                 891|   891|    714|               891|                891|                891|               891|               204|  889|
|   mean|      446.0| 0.3838383838383838| 2.308641975308642|                NULL|  NULL|29.699117647058826|0.5230078563411896|0.38159371492704824|260318.54916792738| 32.2042079685746|              NULL|  NULL|
| stddev|257.3538420152301|0.48659245426485753|0.8360712409770491|                NULL|  NULL|14.526497332334035|1.1027434322934315| 0.8060572211299488|471609.26868834975|49.693428597180897|              NULL|  NULL|
|    min|          1|                  0|                  1|"Andersson, Mr. A...|female|   0.42|                 0|                  0|                 110152|               0.0|               A10|     C|
|    25%|        223|                  0|                  2|                NULL|  NULL|   20.0|                 0|                  0|                19996.0|            7.8958|              NULL|  NULL|
```

```
|   50%|          446|          0|          3|            NULL|  NULL|          28.0|          0|          0|        236171.0|        14.4542| NULL|    NULL|
|   75%|          669|          1|          3|            NULL|  NULL|          38.0|          1|          0|        347743.0|           31.0| NULL|    NULL|
|   max|          891|          1|          3|van Melkebeke, Mr...|  male|          80.0|          8|          6|          WE/P 5735|       512.3292|    T|       S|
+------+-------------+-----------+-----------+--------------------+------+--------------+-----------+-----------+--------------------+---------------+-----+--------+
```

همانطور که می بیند `Embarked` و `Age` خانه خالی دارند. ترکیب `Embarked` به شکل زیر است:

```
+--------+-----+
|Embarked|count|
+--------+-----+
|       Q|   77|
|    NULL|    2|
|       C|  168|
|       S|  644|
+--------+-----+
```

با این که می توان آنها را حذف کرد ولی چون بیشتر مسافران S هستند آنها هم S می گذاریم. در باره سن هم می توانیم آن را با میانگین یا میانه جایگزین کنیم و تفاوت چندانی ندارد (۲۹ و ۲۸) چون چولگی زیادی نداریم. در نهایت این کار را انجام می دهیم.

```
raw_data = raw_data.fillna({
   'Embarked': 'S',
   'Age': 28
})
```

قبل از آموزش مدل توجه کنید که `Sex` و `Embarked` کیفی اسمی هستند. بنابراین باید آنها را با one hot کد کنیم اما جنسیت چون دوتا هست می توانیم به صورت بولی نگهداری کنیم. در گام بعد تمام ویژگی های گفته شده را در ستون ویژگی تجمیع می کنیم. خط لوله را ایجاد می کنیم و مدل را آموزش می دهیم.

```
lg = LogisticRegression(featuresCol='Features', labelCol='Survived')
pipe = Pipeline(
   stages=[
       feature.StringIndexer(inputCol='Embarked',outputCol='Embarked_SI'),

feature.OneHotEncoder(inputCol='Embarked_SI',outputCol='Embarked_OHE'),

       feature.StringIndexer(inputCol='Sex',outputCol='Sex_SI'),

       feature.VectorAssembler(
           inputCols=['Pclass', 'Sex_SI', 'Age', 'Fare', 'Embarked_OHE'],
           outputCol='features'
           ),
       lg
   ]
)

train, test = raw_data.randomSplit([0.8, 0.2])
model = pipe.fit(train)
```

بخش ۴

در این مرحله مدل را ارزیابی می کنیم.

```
test_predict = model.transform(test)
test_predict.select(['Survived', 'prediction']).show(5)

acc = MulticlassClassificationEvaluator(
   labelCol='Survived', predictionCol='prediction', metricName='accuracy'
   ).evaluate(test_predict)
print(f"accuracy: {acc:.4f}")
```

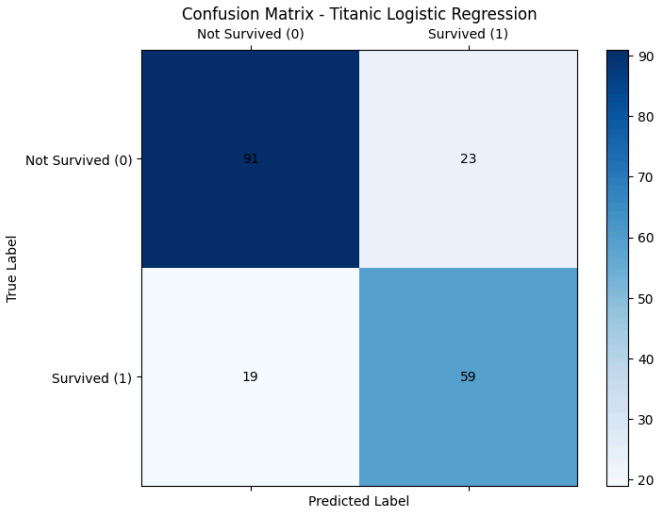
خروجی:

```
+--------+----------+
|Survived|prediction|
+--------+----------+
|       1|       1.0|
|       0|       0.0|
|       0|       1.0|
|       1|       1.0|
|       1|       1.0|
+--------+----------+
only showing top 5 rows

accuracy: 0.7812
```

همانطور که می بینید دقت مدل ۷۸ درصد است.

همچنین ماتریس درهم ریختگی را رسم می کنیم.



در گام آخر هم نتیجه را در یک فایل می نویسم و در hdfs قرار می دهیم.

```
with open('accuracy.txt', "w") as f:
   f.write(f'{acc}')
%%bash
hdfs dfs -put accuracy.txt /vajhi/
```

```
echo '************************'
hdfs dfs -ls /vajhi
echo '************************'
```

```
************************
Found 3 items
-rw-r--r--   1 vajhi dmls      61194 2025-12-26 14:21 /vajhi/Titanic_Dataset.csv
-rw-r--r--   1 vajhi dmls          7 2025-12-26 14:24 /vajhi/accuracy.txt
-rw-r--r--   1 vajhi dmls   52460705 2025-12-25 15:44 /vajhi/wiki_corpus
************************
```