Задача. Повторить сделанное на 8 уроке с другим датасетом.
Решение. Использовал датасет содержащий данные по обследованию пациентов на предмет рака груди в Висконсине.(https://www.kaggle.com/uciml/breast-cancer-wisconsin-data). На выходе происходит чтение батча из кафки с данными пациента, заранее обученная модель ставит диагноз, а также происходит поиск записи об этом пациенте в кассандре, если дубликат найден, то выдается сообщение об этом.

1. Запустил Spark. Загрузил два датасета и потом сделал join для них. Первый содержит id и некоторые данные пациента, а второй содержит id и диагноз пациента. M — злокачественная опухоль, B – доброкачественная опухоль.

```
      ____              __
     / __/__  ___ _____/ /__
    _\ \/ _ \/ _ `/ __/  '_/
   /__ / .__/\_,_/_/ /_/\_\   version 2.4.7
      /_/

Using Python version 2.7.5 (default, Apr  2 2020 13:16:51)
SparkSession available as 'spark'.
>>> from pyspark.ml import Pipeline, PipelineModel
from pyspark.sql import SparkSession, DataFrame
from pyspark.sql.types import StructType, StringType, IntegerType, TimestampType
from pyspark.sql import functions as F
from pyspark.ml.classification import LogisticRegression
from pyspark.ml.feature import OneHotEncoderEstimator, VectorAssembler, CountVectorize
r, StringIndexer, IndexToString>>> from pyspark.sql import SparkSession, DataFrame
>>> from pyspark.sql.types import StructType, StringType, IntegerType, TimestampType
>>> from pyspark.sql import functions as F
>>> from pyspark.ml.classification import LogisticRegression
>>> from pyspark.ml.feature import OneHotEncoderEstimator, VectorAssembler, CountVecto
rizer, StringIndexer, IndexToString
>>> df1 = spark.read.load("id_data_csv",
...                       format="csv", sep=",", inferSchema="true", header="true")
21/01/12 11:36:11 WARN shortcircuit.DomainSocketFactory: The short-circuit local reads
 feature cannot be used because libhadoop cannot be loaded.
>>>
>>> df2 = spark.read.load("id_diag_csv",
...                       format="csv", sep=",", inferSchema="true", header="true")
>>> df1.createOrReplaceTempView("id_data")
21/01/12 11:37:35 WARN util.Utils: Truncated the string representation of a plan since
 it was too large. This behavior can be adjusted by setting 'spark.debug.maxToStringFi
elds' in SparkEnv.conf.
>>> df2.createOrReplaceTempView("id_diag")
>>> spark.sql("select * from id_data").show(10, False)
+--------+-----------+------------+--------------+---------+---------------+----------
------+--------------+-------------------+-------------+----------------------+-------
--+----------+------------+-------+-------------+--------------+------------+---------
--------+-----------+--------------------+------------+-------------+---------------+-
---------+-----------------+-----------------+---------------+--------------------+----
----------+----------------------+
|id      |radius_mean|texture_mean|perimeter_mean|area_mean|smoothness_mean|compactnes
s_mean|concavity_mean|concave points_mean|symmetry_mean|fractal_dimension_mean|radius_
se|texture_se|perimeter_se|area_se|smoothness_se|compactness_se|concavity_se|concave p
oints_se|symmetry_se|fractal_dimension_se|radius_worst|texture_worst|perimeter_worst|a
rea_worst|smoothness_worst|compactness_worst|concavity_worst|concave points_worst|symm
etry_worst|fractal_dimension_worst|
+--------+-----------+------------+--------------+---------+---------------+----------
------+--------------+-------------------+-------------+----------------------+-------
--+----------+------------+-------+-------------+--------------+------------+---------
--------+-----------+--------------------+------------+-------------+---------------+-
---------+-----------------+-----------------+---------------+--------------------+----
----------+----------------------+
|842302  |17.99      |10.38       |122.8         |1001.0   |0.1184         |0.2776
      |0.3001        |0.1471             |0.2419       |0.07871               |1.095
  |0.9053    |8.589       |153.4  |0.006399     |0.04904       |0.05373     |0.01587
        |0.03003    |0.006193           |25.38       |17.33        |184.6          |2
019.0    |0.1622          |0.6656           |0.7119         |0.2654              |0.46
01        |0.1189                 |
|842517  |20.57      |17.77       |132.9         |1326.0   |0.08474        |0.07864
```

```
>>> spark.sql("select * from id_diag").show(10, False)
+--------+---------+
|id      |diagnosis|
+--------+---------+
|842302  |M        |
|842517  |M        |
|84300903|M        |
|84348301|M        |
|84358402|M        |
|843786  |M        |
|844359  |M        |
|84458202|M        |
|844981  |M        |
|84501001|M        |
+--------+---------+
```

```
>>> patients_known = spark.sql("""
... select *
... from id_data
... join id_diag
... where id_data.id = id_diag.id """)
```

2. Использовал модель LR как на занятии но для своих данных.

```
categoricalColumns = []
stages = []
for categoricalCol in categoricalColumns:
    stringIndexer = StringIndexer(inputCol=categoricalCol, outputCol=categoricalCol + 'Index').setHandleInvalid("keep"
    encoder = OneHotEncoderEstimator(inputCols=[stringIndexer.getOutputCol()],
                                     outputCols=[categoricalCol + "classVec"]).setHandleInvalid("keep")
    stages += [stringIndexer, encoder]

label_stringIdx = StringIndexer(inputCol='diagnosis', outputCol='label').setHandleInvalid("keep")
stages += [label_stringIdx]

numericCols = ['radius_mean', 'texture_mean', 'perimeter_mean', 'area_mean', 'smoothness_mean', 'compactness_mean',
               'concavity_mean',
               'concave points_mean', 'symmetry_mean', 'fractal_dimension_mean', 'radius_se', 'texture_se',
               'perimeter_se',
               'area_se', 'smoothness_se', 'compactness_se', 'concavity_se', 'concave points_se', 'symmetry_se',
               'fractal_dimension_se', 'radius_worst', 'texture_worst', 'perimeter_worst', 'area_worst',
               'smoothness_worst', 'compactness_worst', 'concavity_worst', 'concave points_worst', 'symmetry_worst',
               'fractal_dimension_worst']

assemblerInputs = [c + "classVec" for c in categoricalColumns] + numericCols
assembler = VectorAssembler(inputCols=assemblerInputs, outputCol="features").setHandleInvalid("keep")
stages += [assembler]

lr = LogisticRegression(featuresCol='features', labelCol='label', maxIter=10)
stages += [lr]

label_stringIdx_fit = label_stringIdx.fit(patients_known)
indexToStringEstimator = IndexToString().setInputCol("prediction").setOutputCol("category").setLabels(
    label_stringIdx_fit.labels)

stages += [indexToStringEstimator]

pipeline = Pipeline().setStages(stages)
pipelineModel = pipeline.fit(patients_known)

# сохраняем модель на HDFS
pipelineModel.write().overwrite().save("my_LR_model_patients")
```

3. Проверил, что модель работает.

```
>>> pipelineModel.transform(patients_known).select("diagnosis", "category").show(
...     100)
+---------+--------+
|diagnosis|category|
+---------+--------+
|        M|       M|
|        M|       M|
|        M|       M|
|        M|       M|
|        M|       M|
|        M|       M|
|        M|       M|
|        M|       B|
```

4. Создал в kafka свой топик для данных на которых надо будет делать предсказания. И загрузил туда данные из заранее подготовленного csv файла.

```
[BD_243_pstroganov@bigdataanalytics-worker-0 ~]$ /usr/hdp/3.1.4.0-315/kafka/bin/kafka-topics.sh --create
--topic patients_json2 --zookeeper bigdataanalytics-worker-0.novalocal:2181 --partitions 3 --replication-
factor 2 --config retention.ms=-1
WARNING: Due to limitations in metric names, topics with a period ('.') or underscore ('_') could collide
. To avoid issues it is best to use either, but not both.
Created topic "patients_json2".
```

```
>>> raw_files = spark \
...     .readStream \
...     .format("csv") \
...     .schema(schema) \
...     .options(path="csv_stream", header=True) \
...     .load()
>>> out = console_output(raw_files, 5)
>>> -------------------------------------------
Batch: 0
-------------------------------------------
+--------+-----------+------------+--------------+---------+---------------+----------------+------------
--+------------------+-------------+---------------------+---------+----------+------------+-------+---
----------+--------------+------------+-----------------+-----------+--------------------+------------+--
----------+---------------+----------+----------------+----------------+---------------+---------------
-----+--------------+-----------------------+
|      id|radius_mean|texture_mean|perimeter_mean|area_mean|smoothness_mean|compactness_mean|concavity_me
an|concave points_mean|symmetry_mean|fractal_dimension_mean|radius_se|texture_se|perimeter_se|area_se|smo
othness_se|compactness_se|concavity_se|concave points_se|symmetry_se|fractal_dimension_se|radius_worst|te
xture_worst|perimeter_worst|area_worst|smoothness_worst|compactness_worst|concavity_worst|concave points_
worst|symmetry_worst|fractal_dimension_worst|
+--------+-----------+------------+--------------+---------+---------------+----------------+------------
--+------------------+-------------+---------------------+---------+----------+------------+-------+---
----------+--------------+------------+-----------------+-----------+--------------------+------------+--
----------+---------------+----------+----------------+----------------+---------------+---------------
-----+--------------+-----------------------+
|  904357|       11.8|       17.26|         75.26|    431.9|        0.09087|         0.06232|       0.028
53|           0.01638|        0.1847|              0.06019|   0.3438|      1.14|       2.225|  25.06|
  0.005463|       0.01964|     0.02079|         0.005398|    0.01477|           0.003071|       13.45|
      24.49|           86.0|     562.0|          0.1244|           0.1726|         0.1449|              0.
05356|        0.2779|                0.08121|
|90439701|      17.91|       21.02|         124.4|    994.0|          0.123|          0.2576|        0.31
89|           0.1198|        0.2113|              0.07115|    0.403|    0.7747|       3.123|  41.51|
  0.007159|       0.03718|     0.06165|         0.01051|    0.01591|           0.005099|       20.8|
      27.78|          149.6|    1304.0|          0.1873|           0.5917|         0.9034|              0
.1964|        0.3245|                0.1198|
|  904647|      11.93|       10.91|         76.14|    442.7|        0.08872|         0.05242|       0.026
06|           0.01796|        0.1601|              0.05541|   0.2522|     1.045|       1.649|  18.95|
  0.006175|       0.01204|     0.01376|         0.005832|    0.01096|           0.001857|       13.8|
      20.14|          87.64|     589.5|          0.1374|           0.1575|         0.1514|              0.
06876|         0.246|                0.07262|
```

```
>>> def kafka_sink_json(df, freq):
...     return df.selectExpr("CAST(null AS STRING) as key", "CAST(to_json(struct(*)) AS STRING) as value"
) \
...         .writeStream \
...         .format("kafka") \
...         .trigger(processingTime='%s seconds' % freq ) \
...         .option("topic", "patients_json2") \
...         .option("kafka.bootstrap.servers", kafka_brokers) \
...         .option("checkpointLocation", "my_kafka_checkpoint5") \
...         .start()
...
>>> stream = kafka_sink_json(raw_files,5)
>>> patients = spark.readStream. \
...     format("kafka"). \
...     option("kafka.bootstrap.servers", kafka_brokers). \
...     option("subscribe", "patients_json2"). \
...     option("startingOffsets", "earliest"). \
...     option("maxOffsetsPerTrigger", "1"). \
...     load()
>>> s = console_output(patients, 5)
>>> ----------------------------------------
Batch: 0
----------------------------------------
+----+--------------------+--------------+---------+------+--------------------+-------------+
| key|               value|         topic|partition|offset|           timestamp|timestampType|
+----+--------------------+--------------+---------+------+--------------------+-------------+
|null|[7B 22 69 64 22 3...|patients_json2|        1|     0|2021-01-19 12:56:...|            0|
|null|[7B 22 69 64 22 3...|patients_json2|        2|     0|2021-01-19 12:56:...|            0|
|null|[7B 22 69 64 22 3...|patients_json2|        0|     0|2021-01-19 12:56:...|            0|
+----+--------------------+--------------+---------+------+--------------------+-------------+
```

5. Парсинг данных и проверка.

```
>>> value_patients = patients.select(F.from_json(F.col("value").cast("String"), schema).alias("value"), "
offset")
>>> patients_flat = value_patients.select(F.col("value.*"), "offset")
>>> s = console_output(patients_flat, 5)
>>> ----------------------------------------
Batch: 0
----------------------------------------
+--------+-----------+------------+--------------+---------+--------------+----------------+------------
--+------------------+-------------+----------------------+---------+----------+-----------+-------+---
----------+--------------+------------+-----------------+-----------+------------------+------------+--
----------+---------------+-----------+----------------------+------+
|      id|radius_mean|texture_mean|perimeter_mean|area_mean|smoothness_mean|compactness_mean|concavity_me
an|concave points_mean|symmetry_mean|fractal_dimension_mean|radius_se|texture_se|perimeter_se|area_se|smo
othness_se|compactness_se|concavity_se|concave points_se|symmetry_se|fractal_dimension_se|radius_worst|te
xture_worst|perimeter_worst|area_worst|smoothness_worst|compactness_worst|concavity_worst|concave points_
worst|symmetry_worst|fractal_dimension_worst|offset|
+--------+-----------+------------+--------------+---------+--------------+----------------+------------
--+------------------+-------------+----------------------+---------+----------+-----------+-------+---
----------+--------------+------------+-----------------+-----------+------------------+------------+--
----------+---------------+-----------+----------------------+------+
|  904647|      11.93|       10.91|         76.14|    442.7|        0.08872|         0.05242|       0.026
06|           0.01796|        0.1601|               0.05541|   0.2522|     1.045|       1.649|  18.95|
  0.006175|       0.01204|     0.01376|         0.005832|    0.01096|           0.001857|        13.8|
     20.14|         87.64|      589.5|          0.1374|           0.1575|         0.1514|             0.
06876|         0.246|                0.07262|     0|
|90439701|      17.91|       21.02|         124.4|    994.0|          0.123|          0.2576|        0.31
89|           0.1198|        0.2113|               0.07115|    0.403|    0.7747|       3.123|  41.51|
  0.007159|       0.03718|     0.06165|          0.01051|    0.01591|           0.005099|        20.8|
     27.78|         149.6|     1304.0|          0.1873|           0.5917|         0.9034|             0
```

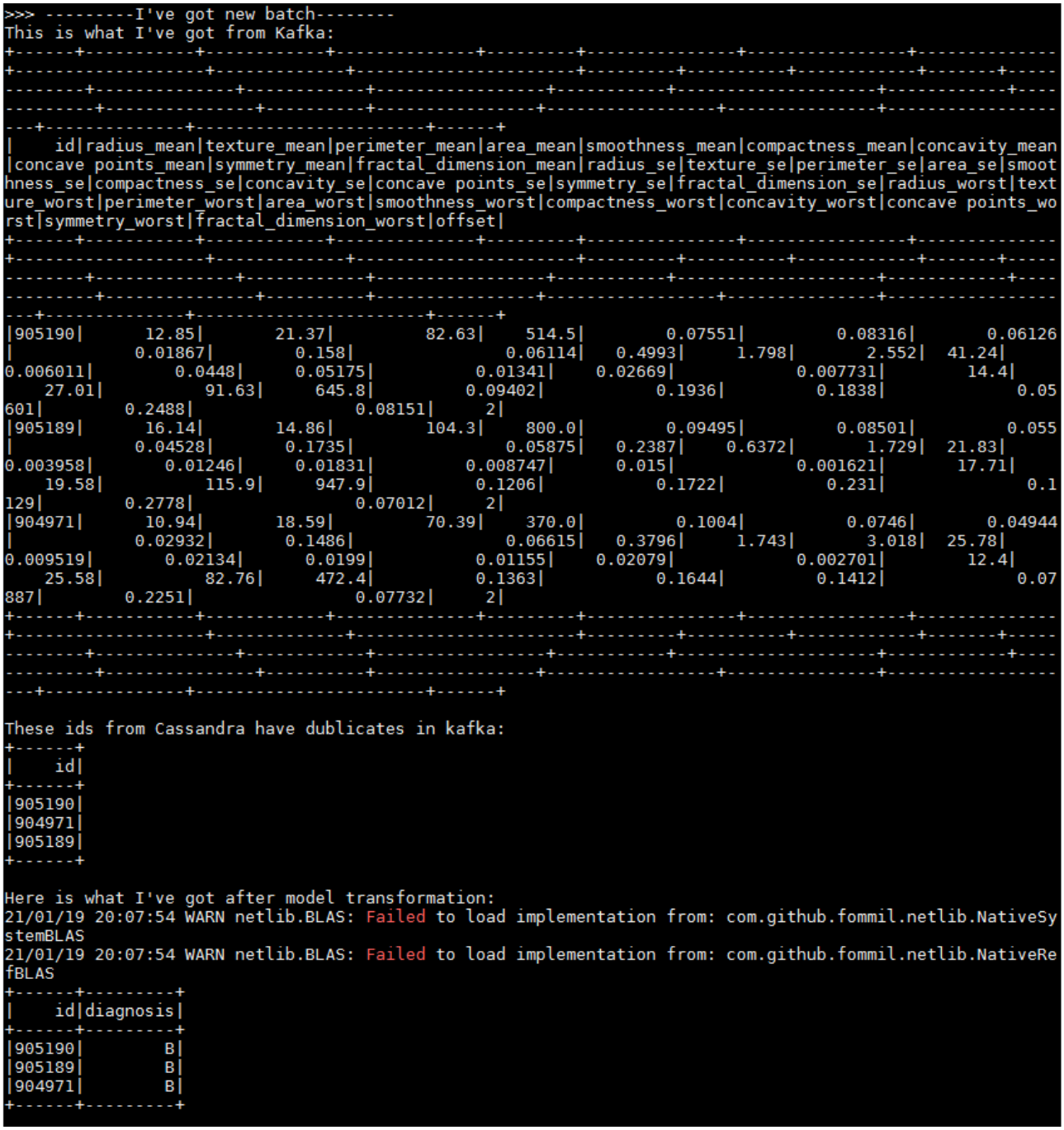
6. Далее создал свой keyspace и таблицу в кассандре и загрузил туда данные из csv файла. Названия колонок с пробелами пришлось переименовать, т.к. создать названия колонок с пробелами в кассандре не удалось (после пробела в описании таблицы она ожидает увидеть название типа). Подробный код можно найти в ру файле.

7. Дальше по аналогии с занятием сделал свою функцию writer_logic и проверил что все работает.

```
def writer_logic(df, epoch_id):
    df.persist()
    print("---------I've got new batch--------")
    print("This is what I've got from Kafka:")
    df.show()
    predict = pipeline_model.transform(df)
    predict_short = predict.select("id", F.col("category").alias("diagnosis"))
    patients_list_df = df.select("id").distinct()
    patients_list_rows = patients_list_df.collect()
    patients_list = map(lambda x: str(x.__getattr__("id")), patients_list_rows)
    where_string = " id = " + " or id = ".join(patients_list)
    print("These ids from Cassandra have dublicates in kafka:")
    dublicate_cass = cass_patients_df.where(where_string)
    dublicate = dublicate_cass.select("id")
    dublicate.show()
    print("Here is what I've got after model transformation:")
    predict_short.show()
    df.unpersist()

#связываем источник Кафки и foreachBatch функцию
stream = patients_flat \
    .writeStream \
    .trigger(processingTime='100 seconds') \
    .foreachBatch(writer_logic) \
    .option("checkpointLocation", "checkpoints/patients_unknown_checkpoint")

#поехали
s = stream.start()
```



8. К данному отчету прилагается py файл с полной последовательностью действий в консоли.