CISC5950 – Big Data Programming – Spring 2022

Professor: Ying Mao
Project 2
Due date: 5/13/2022
Student: Yuliya Akchurina

Data Analysis in Apache Spark

Overview

This project will demonstrate several design examples to analyze data utilizing the Spark programming model running on a three node Hadoop Cluster hosted in Google Cloud. The goal of this project is to practice the use of Spark ML and MLLib libraries to derive meaningful results from the datasets as well as to practice data manipulations in PySpark.

We are going to study the following questions:

P1: Toxic Comment Classification

P2: Heart Disease Prediction using Logistic Regression

P3: Logistic Regression classifier on Census Income Data

P4: Decision Tree and Random Forest on Census Income Data

The following datasets are used for this project:

1. The Toxic Comment Classification dataset, available on Kaggle (link)
2. The Framingham Heart dataset, available on Kaggle (link)
3. The Census Income Data, available on UCI Machine Learning Repository (link)

P1. Toxic Comment Classification

Overview

The goal of this task is to perform text mining on the dataset and to classify each comment into one or more of the predefined classes for the toxic comments based on the contents of the text of the comment. We are going to the Multivariate Logistic Regression Classification model for this task.

Dataset

The dataset is Toxic Comment Classification, publicly available on Kaggle (link). The dataset contains the full text of Wikipedia comments which have been labeled by human raters for toxic behavior. The comments are classified into 6 types of toxicity:

toxic, severe_toxic, obscene, threat, insult, identity_hate
Each comment can belong to zero or all 6 types of toxicity.

The data is split into 3 datasets:
train.csv – containing id, comment text, and labels for the 6 types of toxicity.
test.csv – containing id, comment text
test_labels.csv – containing id, comment labels for the 6 types of toxicity.

Solution

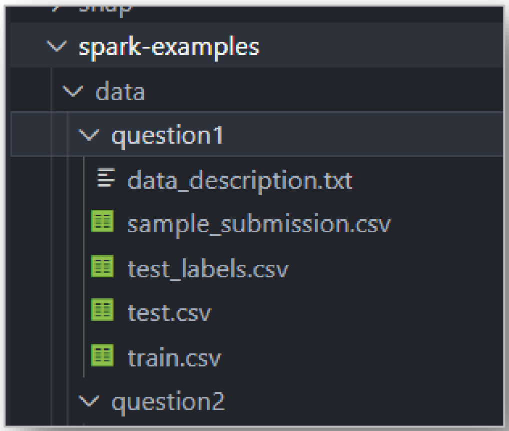
Data Pre-processing:

The dataset is in csv file format. It is located at **/spark-examples/data/question1/**

/spark-examples/data/question1/train.csv

/spark-examples/data/question1/test.csv

/spark-examples/data/question1/test_labels.csv



The **train.csv, test.csv** and **test_labels.csv** are loaded into spark for data cleaning.

To streamline the data cleaning process the three datasets are merged. First **test.csv** and **test_labels.csv** are combined on corresponding column id. Then the resulting dataframe is combined with **train.csv**. The resulting dataframe containing all 3 data files is preprocessed and then split into train and test datasets.

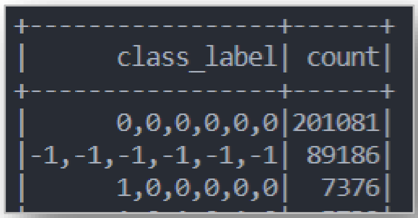
The 6 columns containing classes are combined into one column named ***class_label***. The values correspond to numbers in this order: ***toxic, severe_toxic, obscene, threat, insult, identity_hate***. For example:
0, 0, 0, 0, 0, 0 - non-toxic comment

1, 0, 0, 0, 0, 0 - toxic comment

1, 0, 1, 0, 1, 0 - toxic, obscene, insult

If a value is 0, the label is not assigned, if a value is 1 the label is assigned. There are 64 possible combinations. There are only 41 appearing in this dataset.

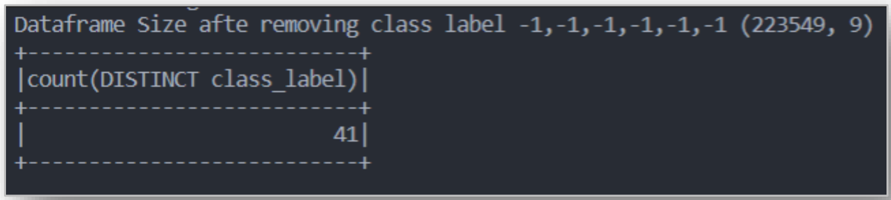
The **test.csv** file had rows with labels "-1" for all 6 categories, with represented comments that were added after the Kaggle competition was completed and therefor these comments were not classified. Since these comments did not have class labels assigned, they were removed from the dataset. In total there were 89,186 rows removed with unclassified comments.



```
+-----------------+------+
|      class_label| count|
+-----------------+------+
|      0,0,0,0,0,0|201081|
|-1,-1,-1,-1,-1,-1| 89186|
|      1,0,0,0,0,0|  7376|
```

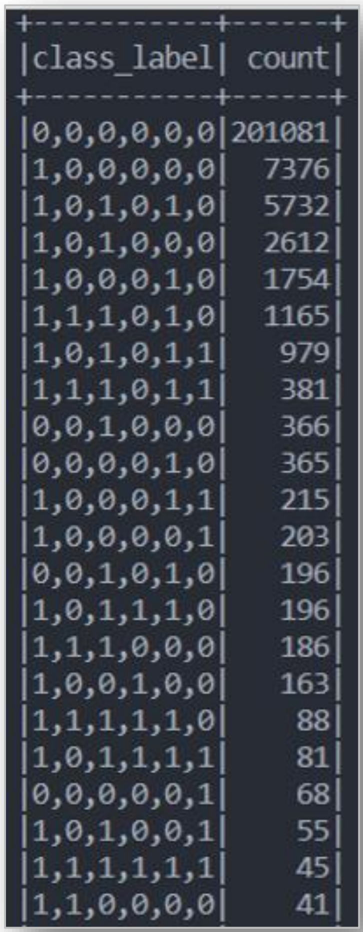
There are 41 resulting distinct class labels.

The resulting combined dataframe has 312,735 rows and 8 columns.

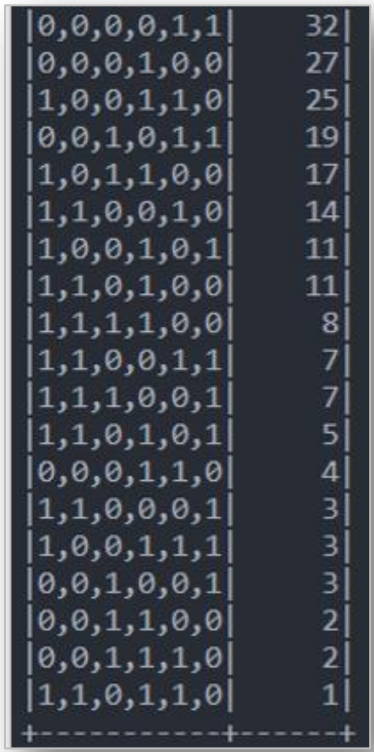


```
Dataframe Size afte removing class label -1,-1,-1,-1,-1,-1 (223549, 9)
+--------------------------+
|count(DISTINCT class_label)|
+--------------------------+
|                        41|
+--------------------------+
```

The count of comments that belongs to each of the 41 label classes



```
+-----------+------+
|class_label| count|
+-----------+------+
|0,0,0,0,0,0|201081|
|1,0,0,0,0,0|  7376|
|1,0,1,0,1,0|  5732|
|1,0,1,0,0,0|  2612|
|1,0,0,0,1,0|  1754|
|1,1,1,0,1,0|  1165|
|1,0,1,0,1,1|   979|
|1,1,1,0,1,1|   381|
|0,0,1,0,0,0|   366|
|0,0,0,0,1,0|   365|
|1,0,0,0,1,1|   215|
|1,0,0,0,0,1|   203|
|0,0,1,0,1,0|   196|
|1,0,1,1,1,0|   196|
|1,1,1,0,0,0|   186|
|1,0,0,1,0,0|   163|
|1,1,1,1,1,0|    88|
|1,0,1,1,1,1|    81|
|0,0,0,0,0,1|    68|
|1,0,1,0,0,1|    55|
|1,1,1,1,1,1|    45|
|1,1,0,0,0,0|    41|
```



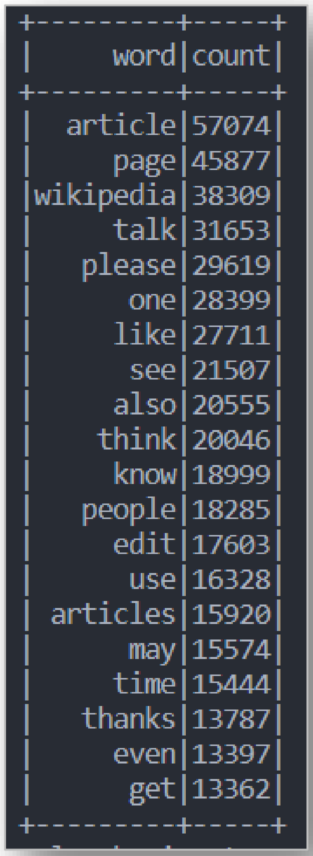
```
|0,0,0,0,1,1|    32|
|0,0,0,1,0,0|    27|
|1,0,0,1,1,0|    25|
|0,0,1,0,1,1|    19|
|1,0,1,1,0,0|    17|
|1,1,0,0,1,0|    14|
|1,0,0,1,0,1|    11|
|1,1,0,1,0,0|    11|
|1,1,1,1,0,0|     8|
|1,1,0,0,1,1|     7|
|1,1,1,0,0,1|     7|
|1,1,0,1,0,1|     5|
|0,0,0,1,1,0|     4|
|1,1,0,0,0,1|     3|
|1,0,0,1,1,1|     3|
|0,0,1,0,0,1|     3|
|0,0,1,1,0,0|     2|
|0,0,1,1,1,0|     2|
|1,1,0,1,1,0|     1|
+-----------+------+
```

Text processing for the ***comment_text*** column:

- Convert to lower case
- Remove new line character "\n"
- Replace word contractions with the correct equivalent
- Remove punctuation and any special characters

- Remove numbers from text
- Remove rows with missing values
- Trim whitespace
- The words are tokenized
- The vectors are created from the text of each comment

Example of the top 20 unique words in the vocabulary with the corresponding counts after the stop words removal.

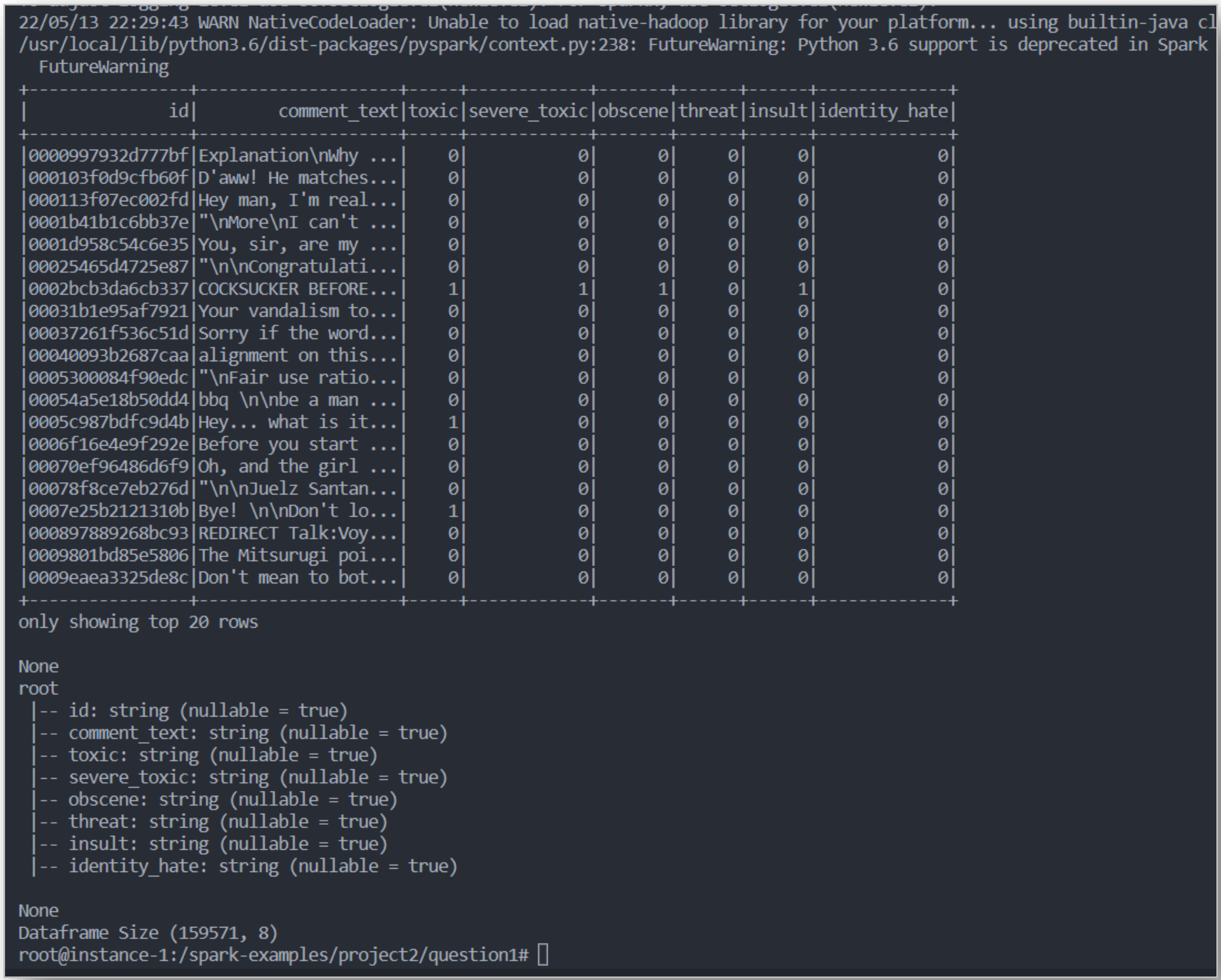


```
+---------+-----+
|     word|count|
+---------+-----+
|  article|57074|
|     page|45877|
|wikipedia|38309|
|     talk|31653|
|   please|29619|
|      one|28399|
|     like|27711|
|      see|21507|
|     also|20555|
|    think|20046|
|     know|18999|
|   people|18285|
|     edit|17603|
|      use|16328|
| articles|15920|
|      may|15574|
|     time|15444|
|   thanks|13787|
|     even|13397|
|      get|13362|
+---------+-----+
```

The ***class_label*** column is indexed. Each of the 41 distinct classes is assigned a number based of the frequency.

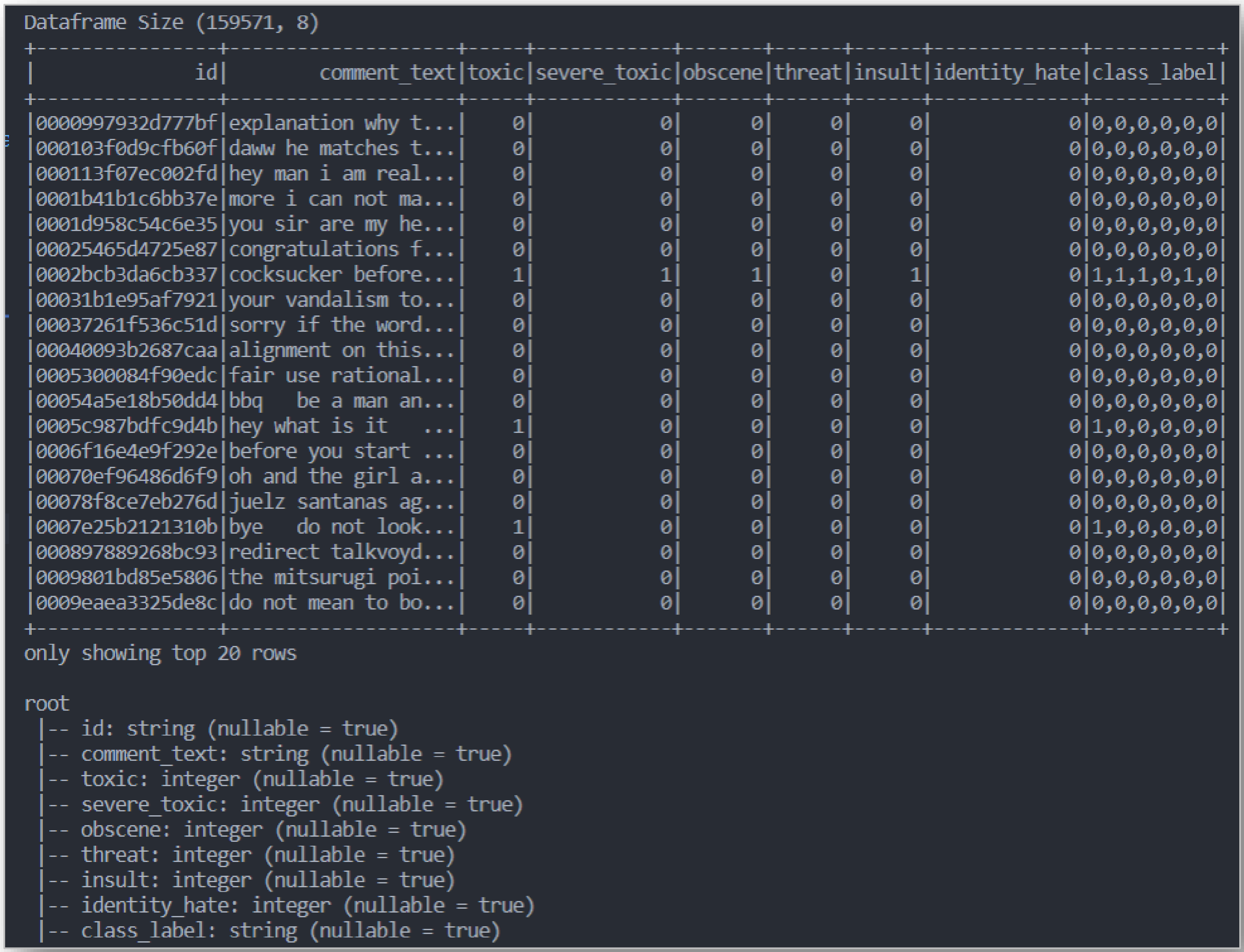
After processing the data is split into train and test datasets 70/30 ratio. Now the data is ready to be used in the Logistic Regression Classifier model

The snapshot of the data before preprocessing

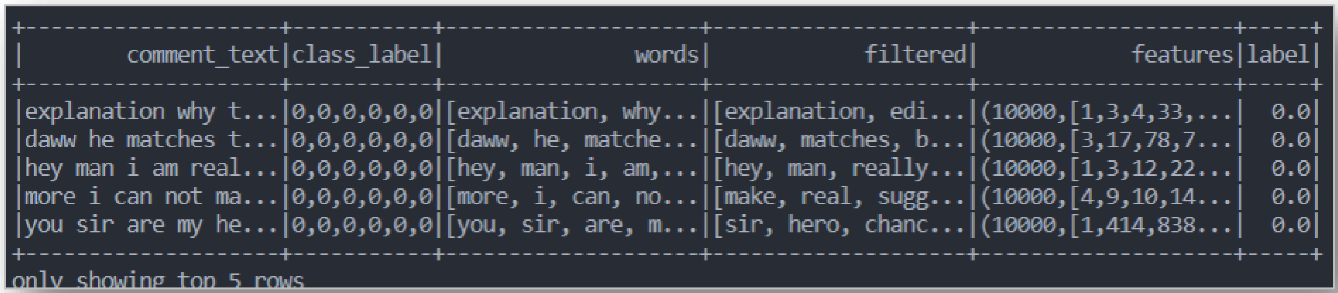


```
22/05/13 22:29:43 WARN NativeCodeLoader: Unable to load native-hadoop library for your platform... using builtin-java cl
/usr/local/lib/python3.6/dist-packages/pyspark/context.py:238: FutureWarning: Python 3.6 support is deprecated in Spark
  FutureWarning
+----------------+--------------------+-----+------------+-------+------+------+-------------+
|              id|        comment_text|toxic|severe_toxic|obscene|threat|insult|identity_hate|
+----------------+--------------------+-----+------------+-------+------+------+-------------+
|0000997932d777bf|Explanation\nWhy ...|    0|           0|      0|     0|     0|            0|
|000103f0d9cfb60f|D'aww! He matches...|    0|           0|      0|     0|     0|            0|
|000113f07ec002fd|Hey man, I'm real...|    0|           0|      0|     0|     0|            0|
|0001b41b1c6bb37e|"\nMore\nI can't ...|    0|           0|      0|     0|     0|            0|
|0001d958c54c6e35|You, sir, are my ...|    0|           0|      0|     0|     0|            0|
|00025465d4725e87|"\n\nCongratulati...|    0|           0|      0|     0|     0|            0|
|0002bcb3da6cb337|COCKSUCKER BEFORE...|    1|           1|      1|     0|     1|            0|
|00031b1e95af7921|Your vandalism to...|    0|           0|      0|     0|     0|            0|
|00037261f536c51d|Sorry if the word...|    0|           0|      0|     0|     0|            0|
|00040093b2687caa|alignment on this...|    0|           0|      0|     0|     0|            0|
|0005300084f90edc|"\nFair use ratio...|    0|           0|      0|     0|     0|            0|
|00054a5e18b50dd4|bbq \n\nbe a man ...|    0|           0|      0|     0|     0|            0|
|0005c987bdfc9d4b|Hey... what is it...|    1|           0|      0|     0|     0|            0|
|0006f16e4e9f292e|Before you start ...|    0|           0|      0|     0|     0|            0|
|00070ef96486d6f9|Oh, and the girl ...|    0|           0|      0|     0|     0|            0|
|00078f8ce7eb276d|"\n\nJuelz Santan...|    0|           0|      0|     0|     0|            0|
|0007e25b2121310b|Bye! \n\nDon't lo...|    1|           0|      0|     0|     0|            0|
|000897889268bc93|REDIRECT Talk:Voy...|    0|           0|      0|     0|     0|            0|
|0009801bd85e5806|The Mitsurugi poi...|    0|           0|      0|     0|     0|            0|
|0009eaea3325de8c|Don't mean to bot...|    0|           0|      0|     0|     0|            0|
+----------------+--------------------+-----+------------+-------+------+------+-------------+
only showing top 20 rows

None
root
 |-- id: string (nullable = true)
 |-- comment_text: string (nullable = true)
 |-- toxic: string (nullable = true)
 |-- severe_toxic: string (nullable = true)
 |-- obscene: string (nullable = true)
 |-- threat: string (nullable = true)
 |-- insult: string (nullable = true)
 |-- identity_hate: string (nullable = true)

None
Dataframe Size (159571, 8)
root@instance-1:/spark-examples/project2/question1# 
```

After pre-processing



```
Dataframe Size (159571, 8)
+----------------+--------------------+-----+------------+-------+------+------+-------------+-----------+
|              id|        comment_text|toxic|severe_toxic|obscene|threat|insult|identity_hate|class_label|
+----------------+--------------------+-----+------------+-------+------+------+-------------+-----------+
|0000997932d777bf|explanation why t...|    0|           0|      0|     0|     0|            0|0,0,0,0,0,0|
|000103f0d9cfb60f|daww he matches t...|    0|           0|      0|     0|     0|            0|0,0,0,0,0,0|
|000113f07ec002fd|hey man i am real...|    0|           0|      0|     0|     0|            0|0,0,0,0,0,0|
|0001b41b1c6bb37e|more i can not ma...|    0|           0|      0|     0|     0|            0|0,0,0,0,0,0|
|0001d958c54c6e35|you sir are my he...|    0|           0|      0|     0|     0|            0|0,0,0,0,0,0|
|00025465d4725e87|congratulations f...|    0|           0|      0|     0|     0|            0|0,0,0,0,0,0|
|0002bcb3da6cb337|cocksucker before...|    1|           1|      1|     0|     1|            0|1,1,1,0,1,0|
|00031b1e95af7921|your vandalism to...|    0|           0|      0|     0|     0|            0|0,0,0,0,0,0|
|00037261f536c51d|sorry if the word...|    0|           0|      0|     0|     0|            0|0,0,0,0,0,0|
|00040093b2687caa|alignment on this...|    0|           0|      0|     0|     0|            0|0,0,0,0,0,0|
|0005300084f90edc|fair use rational...|    0|           0|      0|     0|     0|            0|0,0,0,0,0,0|
|00054a5e18b50dd4|bbq   be a man an...|    0|           0|      0|     0|     0|            0|0,0,0,0,0,0|
|0005c987bdfc9d4b|hey what is it   ...|    1|           0|      0|     0|     0|            0|1,0,0,0,0,0|
|0006f16e4e9f292e|before you start ...|    0|           0|      0|     0|     0|            0|0,0,0,0,0,0|
|00070ef96486d6f9|oh and the girl a...|    0|           0|      0|     0|     0|            0|0,0,0,0,0,0|
|00078f8ce7eb276d|juelz santanas ag...|    0|           0|      0|     0|     0|            0|0,0,0,0,0,0|
|0007e25b2121310b|bye   do not look...|    1|           0|      0|     0|     0|            0|1,0,0,0,0,0|
|000897889268bc93|redirect talkvoyd...|    0|           0|      0|     0|     0|            0|0,0,0,0,0,0|
|0009801bd85e5806|the mitsurugi poi...|    0|           0|      0|     0|     0|            0|0,0,0,0,0,0|
|0009eaea3325de8c|do not mean to bo...|    0|           0|      0|     0|     0|            0|0,0,0,0,0,0|
+----------------+--------------------+-----+------------+-------+------+------+-------------+-----------+
only showing top 20 rows

root
 |-- id: string (nullable = true)
 |-- comment_text: string (nullable = true)
 |-- toxic: integer (nullable = true)
 |-- severe_toxic: integer (nullable = true)
 |-- obscene: integer (nullable = true)
 |-- threat: integer (nullable = true)
 |-- insult: integer (nullable = true)
 |-- identity_hate: integer (nullable = true)
 |-- class_label: string (nullable = true)
```



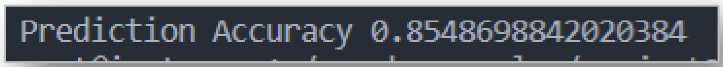
```
+--------------------+-----------+--------------------+--------------------+--------------------+-----+
|        comment_text|class_label|               words|            filtered|            features|label|
+--------------------+-----------+--------------------+--------------------+--------------------+-----+
|explanation why t...|0,0,0,0,0,0|[explanation, why...|[explanation, edi...|(10000,[1,3,4,33,...|  0.0|
|daww he matches t...|0,0,0,0,0,0|[daww, he, matche...|[daww, matches, b...|(10000,[3,17,78,7...|  0.0|
|hey man i am real...|0,0,0,0,0,0|[hey, man, i, am,...|[hey, man, really...|(10000,[1,3,12,22...|  0.0|
|more i can not ma...|0,0,0,0,0,0|[more, i, can, no...|[make, real, sugg...|(10000,[4,9,10,14...|  0.0|
|you sir are my he...|0,0,0,0,0,0|[you, sir, are, m...|[sir, hero, chanc...|(10000,[1,414,838...|  0.0|
+--------------------+-----------+--------------------+--------------------+--------------------+-----+
only showing top 5 rows
```

Model and Results:

The model used for prediction is **LogisticRegression** from **pyspark.ml.classification** library.

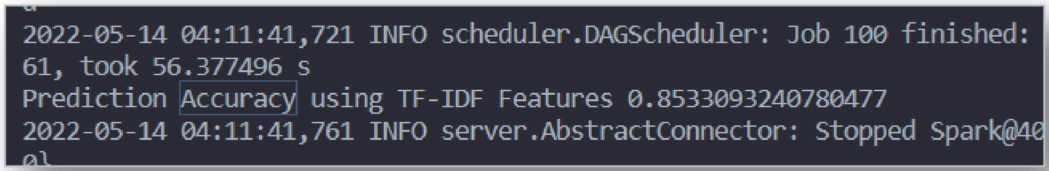
CountVectorizer and **HashingTF** were used to create vectors from the pre-processed text of the comments. The performance of the Logistic regression is compared.

The accuracy of the Logistic regression model with the use **CountVectorizer** is 85.49%.



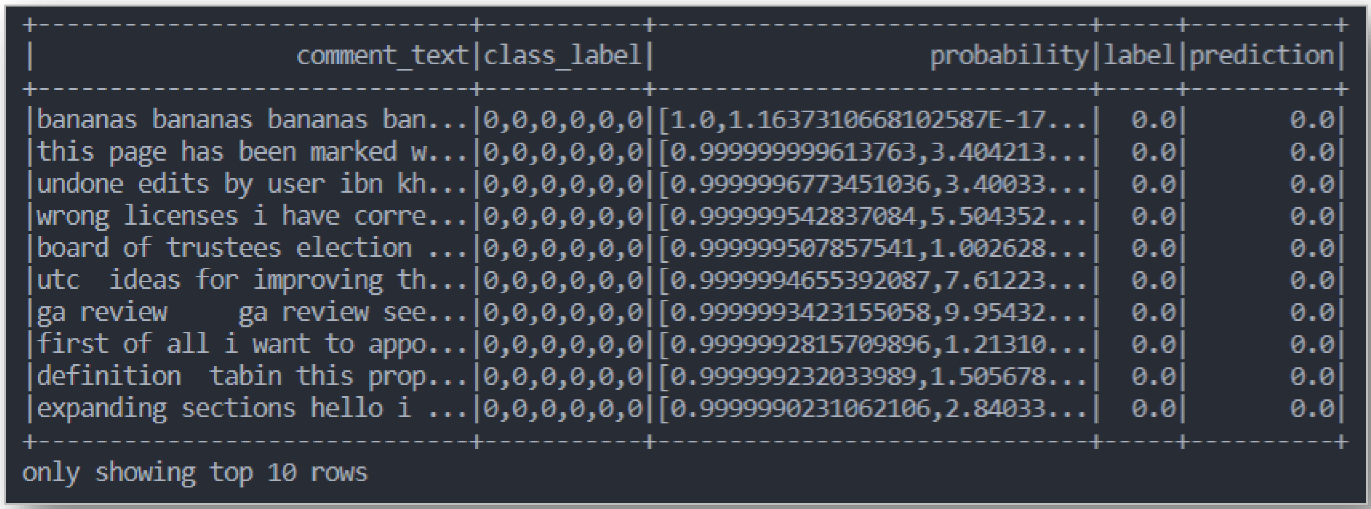
```
Prediction Accuracy 0.8548698842020384
```

The accuracy of the Logistic regression model with the use of **HashingTF** is 85.33%.



```
2022-05-14 04:11:41,721 INFO scheduler.DAGScheduler: Job 100 finished:
61, took 56.377496 s
Prediction Accuracy using TF-IDF Features 0.8533093240780477
2022-05-14 04:11:41,761 INFO server.AbstractConnector: Stopped Spark@40
```

Example of the predictions made by the logistic regression model:



```
+----------------------------+-----------+----------------------------+-----+----------+
|                comment_text|class_label|                 probability|label|prediction|
+----------------------------+-----------+----------------------------+-----+----------+
|bananas bananas bananas ban...|0,0,0,0,0,0|[1.0,1.1637310668102587E-17...|  0.0|       0.0|
|this page has been marked w...|0,0,0,0,0,0|[0.999999999613763,3.404213...|  0.0|       0.0|
|undone edits by user ibn kh...|0,0,0,0,0,0|[0.9999996773451036,3.40033...|  0.0|       0.0|
|wrong licenses i have corre...|0,0,0,0,0,0|[0.999999542837084,5.504352...|  0.0|       0.0|
|board of trustees election ...|0,0,0,0,0,0|[0.999999507857541,1.002628...|  0.0|       0.0|
|utc  ideas for improving th...|0,0,0,0,0,0|[0.9999994655392087,7.61223...|  0.0|       0.0|
|ga review     ga review see...|0,0,0,0,0,0|[0.9999993423155058,9.95432...|  0.0|       0.0|
|first of all i want to appo...|0,0,0,0,0,0|[0.9999992815709896,1.21310...|  0.0|       0.0|
|definition  tabin this prop...|0,0,0,0,0,0|[0.999999232033989,1.505678...|  0.0|       0.0|
|expanding sections hello i ...|0,0,0,0,0,0|[0.9999990231062106,2.84033...|  0.0|       0.0|
+----------------------------+-----------+----------------------------+-----+----------+
only showing top 10 rows
```

P2. Heart Disease Prediction using Logistic Regression

Overview

This part is going to use Logistic Regression model in predict overall risk of developing heart disease based on the features in dataset.

Dataset

The dataset is Framingham Heart dataset (kaggle). The dataset is publicly available on the Kaggle website, and it is from an ongoing cardiovascular study on residents of the town of Framingham, Massachusetts. The classification goal is to predict whether the patient has 10-year risk of future coronary heart disease (CHD). The dataset provides the patients' information. It includes over 4,000 records and 15 attributes.

Variables:

Each attribute is a potential risk factor. There are both demographic, behavioral, and medical risk factors.

Demographic:

- Sex: male or female (Nominal)
- Age: Age of the patient;(Continuous - Although the recorded ages have been truncated to whole numbers, the concept of age is continuous)

Behavioral:

- Current Smoker: whether or not the patient is a current smoker (Nominal)
- Cigs Per Day: the number of cigarettes that the person smoked on average in one day.(can be considered continuous as one can have any number of cigarettes, even half a cigarette.)

Medical (history):

- BP Meds: whether or not the patient was on blood pressure medication (Nominal)
- Prevalent Stroke: whether or not the patient had previously had a stroke (Nominal)
- Prevalent Hyp: whether or not the patient was hypertensive (Nominal)
- Diabetes: whether or not the patient had diabetes (Nominal)

Medical (current):

- Tot Chol: total cholesterol level (Continuous)
- Sys BP: systolic blood pressure (Continuous)
- Dia BP: diastolic blood pressure (Continuous)
- BMI: Body Mass Index (Continuous)
- Heart Rate: heart rate (Continuous - In medical research, variables such as heart rate though in fact discrete, yet are considered continuous because of large number of possible values.)
- Glucose: glucose level (Continuous)

Predict variable (desired target):

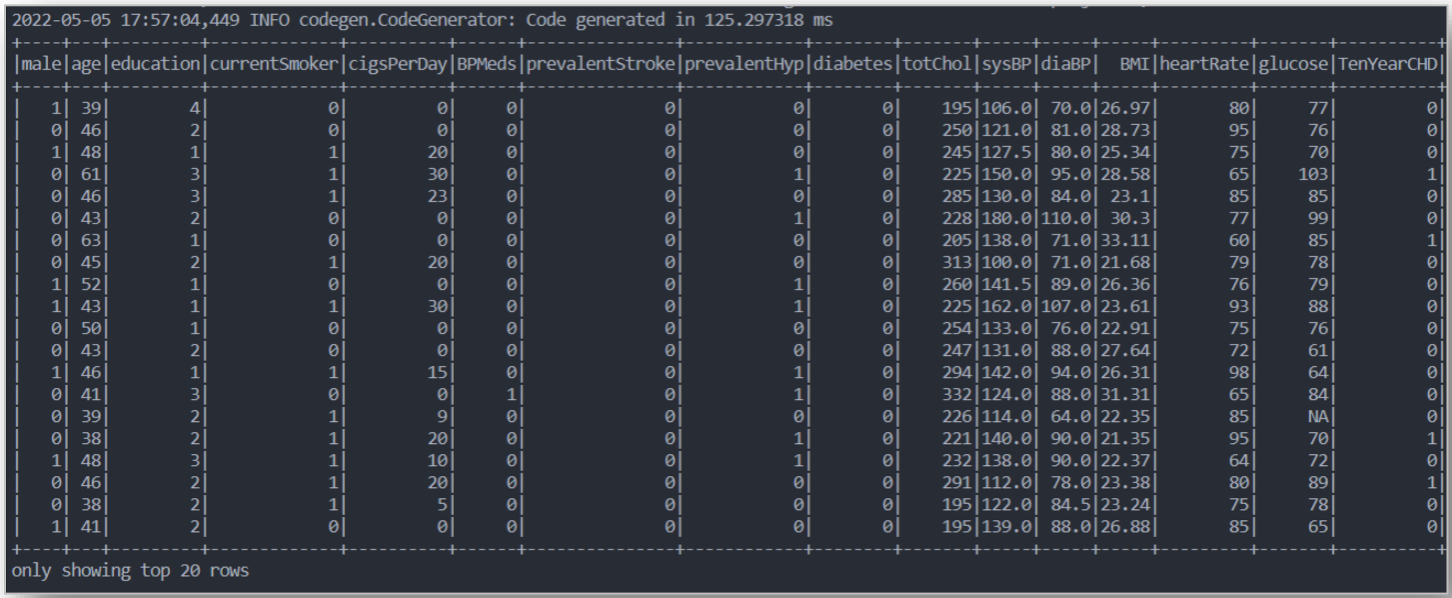
- 10 year risk of coronary heart disease CHD (binary: "1", means "Yes", "0" means "No")

Solution:

Data Pre-processing:

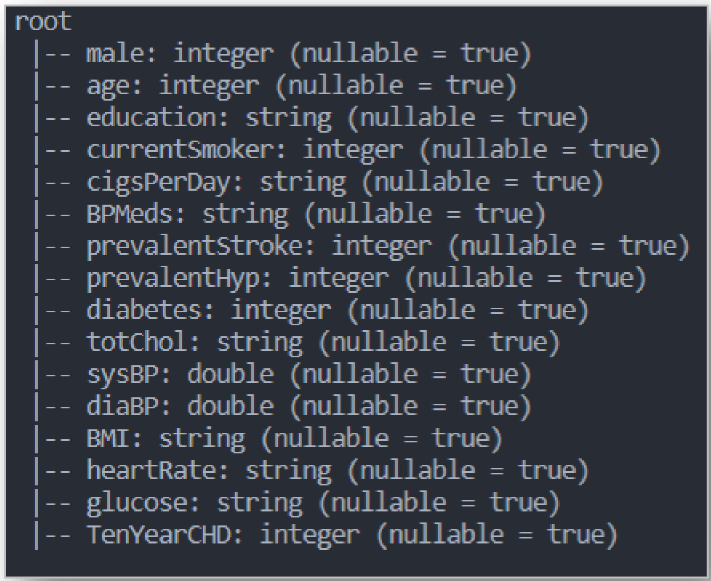
The dataset is in csv file format. It is located at **/spark-examples/data/question2/framingham.csv**

Below is a snapshot of the data uploaded to a dataframe:



```
2022-05-05 17:57:04,449 INFO codegen.CodeGenerator: Code generated in 125.297318 ms
+----+---+---------+-------------+----------+------+---------------+------------+--------+-------+-----+-----+-----+---------+-------+----------+
|male|age|education|currentSmoker|cigsPerDay|BPMeds|prevalentStroke|prevalentHyp|diabetes|totChol|sysBP|diaBP|  BMI|heartRate|glucose|TenYearCHD|
+----+---+---------+-------------+----------+------+---------------+------------+--------+-------+-----+-----+-----+---------+-------+----------+
|   1| 39|        4|            0|         0|     0|              0|           0|       0|    195|106.0| 70.0|26.97|       80|     77|         0|
|   0| 46|        2|            0|         0|     0|              0|           0|       0|    250|121.0| 81.0|28.73|       95|     76|         0|
|   1| 48|        1|            1|        20|     0|              0|           0|       0|    245|127.5| 80.0|25.34|       75|     70|         0|
|   0| 61|        3|            1|        30|     0|              0|           1|       0|    225|150.0| 95.0|28.58|       65|    103|         1|
|   0| 46|        3|            1|        23|     0|              0|           0|       0|    285|130.0| 84.0| 23.1|       85|     85|         0|
|   0| 43|        2|            0|         0|     0|              0|           1|       0|    228|180.0|110.0| 30.3|       77|     99|         0|
|   0| 63|        1|            0|         0|     0|              0|           0|       0|    205|138.0| 71.0|33.11|       60|     85|         1|
|   0| 45|        2|            1|        20|     0|              0|           0|       0|    313|100.0| 71.0|21.68|       79|     78|         0|
|   1| 52|        1|            0|         0|     0|              0|           1|       0|    260|141.5| 89.0|26.36|       76|     79|         0|
|   1| 43|        1|            1|        30|     0|              0|           1|       0|    225|162.0|107.0|23.61|       93|     88|         0|
|   0| 50|        1|            0|         0|     0|              0|           0|       0|    254|133.0| 76.0|22.91|       75|     76|         0|
|   0| 43|        2|            0|         0|     0|              0|           0|       0|    247|131.0| 88.0|27.64|       72|     61|         0|
|   1| 46|        1|            1|        15|     0|              0|           1|       0|    294|142.0| 94.0|26.31|       98|     64|         0|
|   0| 41|        3|            0|         0|     1|              0|           1|       0|    332|124.0| 88.0|31.31|       65|     84|         0|
|   0| 39|        2|            1|         9|     0|              0|           0|       0|    226|114.0| 64.0|22.35|       85|     NA|         0|
|   0| 38|        2|            1|        20|     0|              0|           1|       0|    221|140.0| 90.0|21.35|       95|     70|         1|
|   1| 48|        3|            1|        10|     0|              0|           1|       0|    232|138.0| 90.0|22.37|       64|     72|         0|
|   0| 46|        2|            1|        20|     0|              0|           0|       0|    291|112.0| 78.0|23.38|       80|     89|         1|
|   0| 38|        2|            1|         5|     0|              0|           0|       0|    195|122.0| 84.5|23.24|       75|     78|         0|
|   1| 41|        2|            0|         0|     0|              0|           0|       0|    195|139.0| 88.0|26.88|       85|     65|         0|
+----+---+---------+-------------+----------+------+---------------+------------+--------+-------+-----+-----+-----+---------+-------+----------+
only showing top 20 rows
```

Dataframe schema



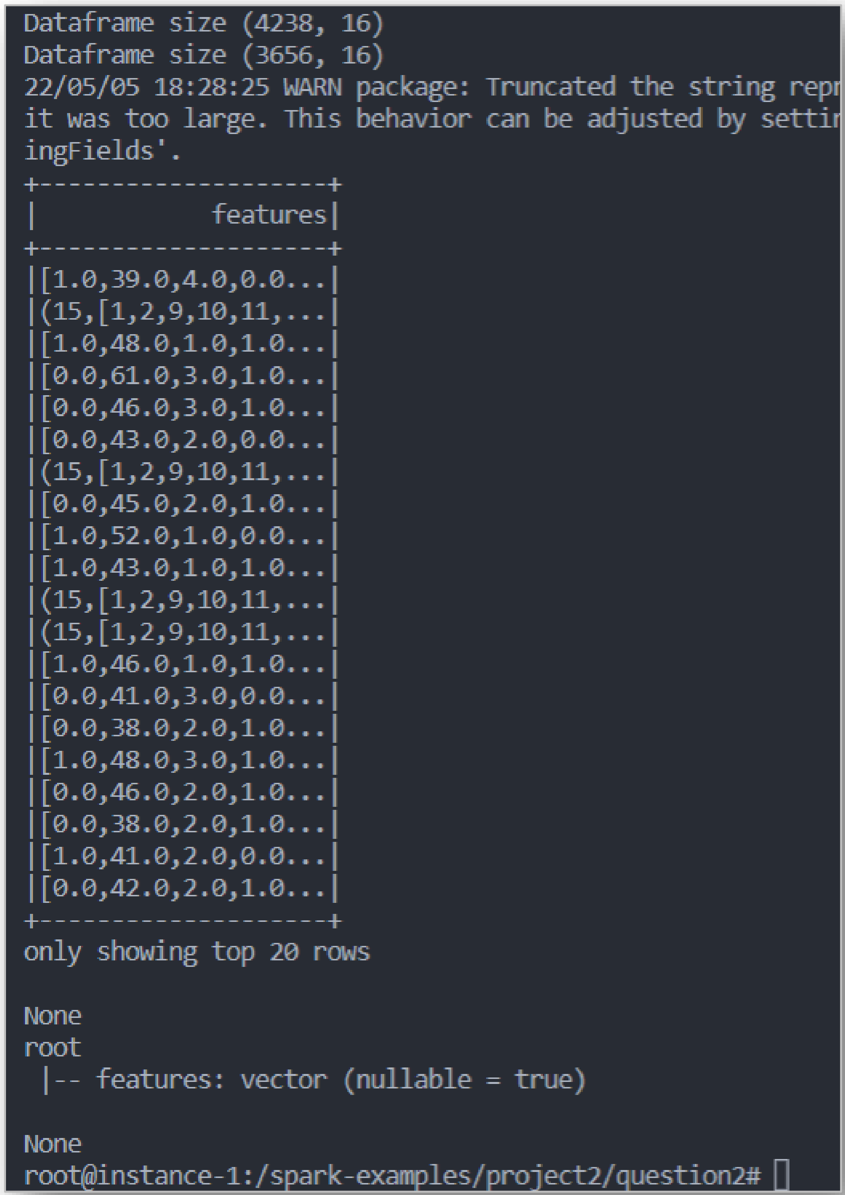
```
root
 |-- male: integer (nullable = true)
 |-- age: integer (nullable = true)
 |-- education: string (nullable = true)
 |-- currentSmoker: integer (nullable = true)
 |-- cigsPerDay: string (nullable = true)
 |-- BPMeds: string (nullable = true)
 |-- prevalentStroke: integer (nullable = true)
 |-- prevalentHyp: integer (nullable = true)
 |-- diabetes: integer (nullable = true)
 |-- totChol: string (nullable = true)
 |-- sysBP: double (nullable = true)
 |-- diaBP: double (nullable = true)
 |-- BMI: string (nullable = true)
 |-- heartRate: string (nullable = true)
 |-- glucose: string (nullable = true)
 |-- TenYearCHD: integer (nullable = true)
```

The steps for data preprocessing:

- Removing rows with missing values
- Convert all columns into double or integer data type to use in the Logistic Regression Model

- Rename the "***TenYearCHD***" column to "***label***"
- Create the feature vector using VectorAssembler function from the 15 attribute columns to use as input to the Logistic Regression Model
- Split dataset 80/20 into training and test data

The snapshot of the vector column created from the 15 attributes called ***features*** column.

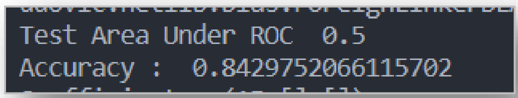


```
Dataframe size (4238, 16)
Dataframe size (3656, 16)
22/05/05 18:28:25 WARN package: Truncated the string repr
it was too large. This behavior can be adjusted by settir
ingFields'.
+--------------------+
|            features|
+--------------------+
|[1.0,39.0,4.0,0.0...|
|(15,[1,2,9,10,11,...|
|[1.0,48.0,1.0,1.0...|
|[0.0,61.0,3.0,1.0...|
|[0.0,46.0,3.0,1.0...|
|[0.0,43.0,2.0,0.0...|
|(15,[1,2,9,10,11,...|
|[0.0,45.0,2.0,1.0...|
|[1.0,52.0,1.0,0.0...|
|[1.0,43.0,1.0,1.0...|
|(15,[1,2,9,10,11,...|
|(15,[1,2,9,10,11,...|
|[1.0,46.0,1.0,1.0...|
|[0.0,41.0,3.0,0.0...|
|[0.0,38.0,2.0,1.0...|
|[1.0,48.0,3.0,1.0...|
|[0.0,46.0,2.0,1.0...|
|[0.0,38.0,2.0,1.0...|
|[1.0,41.0,2.0,0.0...|
|[0.0,42.0,2.0,1.0...|
+--------------------+
only showing top 20 rows

None
root
 |-- features: vector (nullable = true)

None
root@instance-1:/spark-examples/project2/question2#
```

Model and Results:

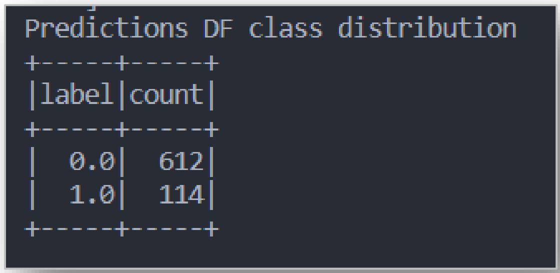
Once the data is ready, it is used as input for the **LogisticRegression** model from the **pyspark.ml.classification** library.

Binary Logistic Regression Model was used since there are two possible classes. The resulting accuracy is 84.30%



```
Test Area Under ROC  0.5
Accuracy :  0.8429752066115702
```

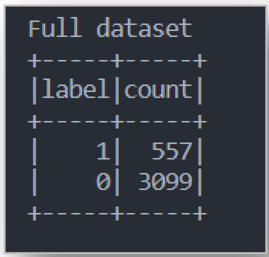
The distribution of the predicted classes:



```
Predictions DF class distribution
+-----+-----+
|label|count|
+-----+-----+
|  0.0|  612|
|  1.0|  114|
+-----+-----+
```

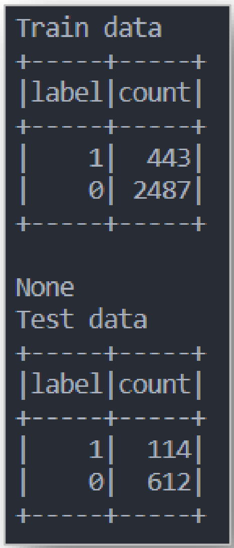
The model prediction accuracy is not as high as expected. This might be due to imbalanced dataset. The data is skewed with higher attribution to one class than the other. The way to improve model performance is to balance the training dataset.

The class distribution in the entire dataset:



```
Full dataset
+-----+-----+
|label|count|
+-----+-----+
|    1|  557|
|    0| 3099|
+-----+-----+
```

The class distribution in the resulting train and test datasets:



```
Train data
+-----+-----+
|label|count|
+-----+-----+
|    1|  443|
|    0| 2487|
+-----+-----+

None
Test data
+-----+-----+
|label|count|
+-----+-----+
|    1|  114|
|    0|  612|
+-----+-----+
```

P3. Logistic Regression classifier on Census Income Data

Overview

This part is going to use Logistic Regression Classifier in Spark ML to process the Census Income Data and predict the income level based of the features in dataset.

Dataset

The dataset is Census Dataset from the UCI Machine Learning Repository (link). The data is split into train and test files. There dataset contains 48842 instances, mix of continuous and discrete (train=32561, test=16281). 45222 if instances with unknown values are removed (train=30162, test=15060). Prediction task is to determine whether a person makes over 50K a year.

Features (the columns in the dataset):

- **age:** continuous.
- **workclass**: Private, Self-emp-not-inc, Self-emp-inc, Federal-gov, Local-gov, State-gov, Without-pay, Never-worked.
- **fnlwgt**: continuous.
- **education**: Bachelors, Some-college, 11th, HS-grad, Prof-school, Assoc-acdm, Assoc-voc, 9th, 7th-8th, 12th, Masters, 1st-4th, 10th, Doctorate, 5th-6th, Preschool.
- **education-num**: continuous.
- **marital-status**: Married-civ-spouse, Divorced, Never-married, Separated, Widowed, Married-spouse-absent, Married-AF-spouse.
- **occupation**: Tech-support, Craft-repair, Other-service, Sales, Exec-managerial, Prof-specialty, Handlers-cleaners, Machine-op-inspct, Adm-clerical, Farming-fishing, Transport-moving, Priv-house-serv, Protective-serv, Armed-Forces.
- **relationship**: Wife, Own-child, Husband, Not-in-family, Other-relative, Unmarried.
- **race**: White, Asian-Pac-Islander, Amer-Indian-Eskimo, Other, Black.
- **sex**: Female, Male.
- **capital-gain**: continuous.
- **capital-loss**: continuous.
- **hours-per-week**: continuous.
- **native-country**: United-States, Cambodia, England, Puerto-Rico, Canada, Germany, Outlying-US(Guam-USVI-etc), India, Japan, Greece, South, China, Cuba, Iran, Honduras, Philippines, Italy, Poland, Jamaica, Vietnam, Mexico, Portugal, Ireland, France, Dominican-Republic, Laos, Ecuador, Taiwan, Haiti, Columbia, Hungary, Guatemala, Nicaragua, Scotland, Thailand, Yugoslavia, El-Salvador, Trinadad&Tobago, Peru, Hong, Holand-Netherlands.

Solution

Data Pre-processing:

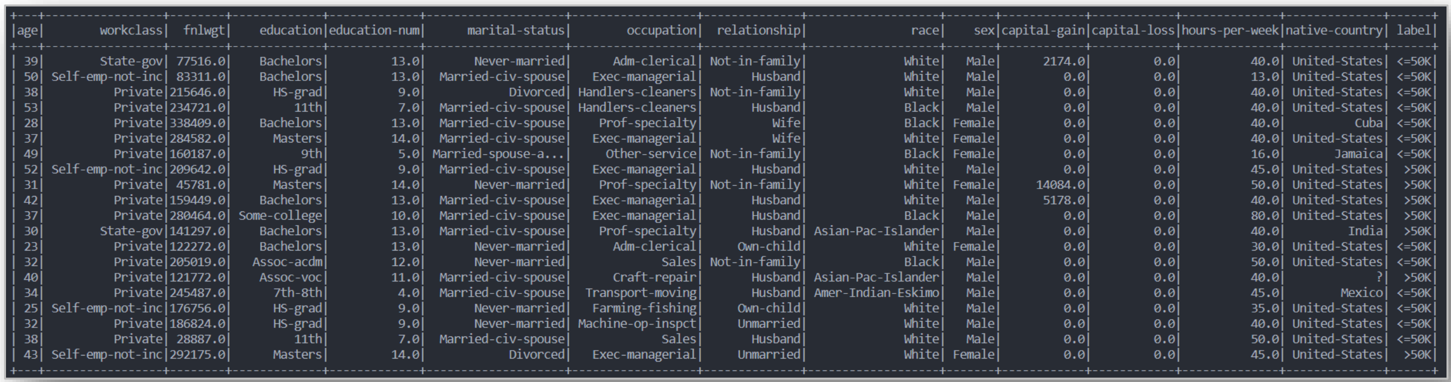
The data is in the csv file located in data folder at:

Train dataset - **/spark-examples/data/question3_4/adult_data.csv**

Test dataset - **/spark-examples/data/question3_4/adult_test.csv**

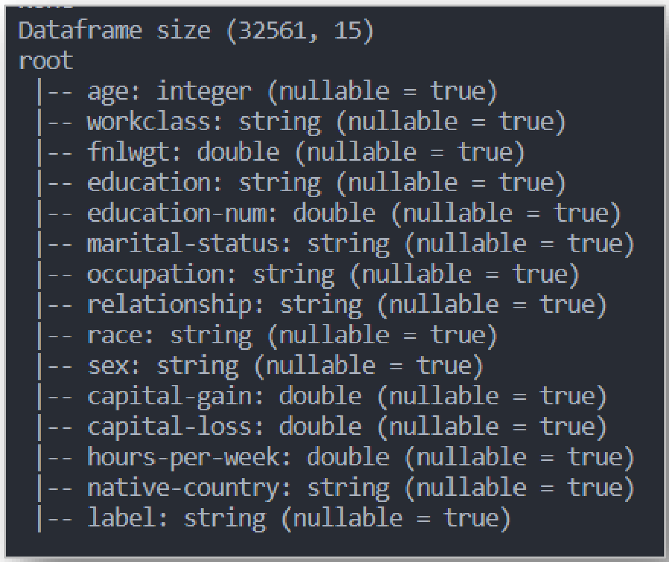
Prior to applying the Logistic Regression model both train and test datasets must be cleaned and pre-processed.

Here is the example of the train dataset uploaded to the dataframe.



```
+---+-----------------+--------+-------------+-------------+--------------------+------------------+--------------+-------------------+-------+------------+------------+--------------+--------------+------+
|age|        workclass|  fnlwgt|    education|education-num|      marital-status|        occupation|  relationship|               race|    sex|capital-gain|capital-loss|hours-per-week|native-country| label|
+---+-----------------+--------+-------------+-------------+--------------------+------------------+--------------+-------------------+-------+------------+------------+--------------+--------------+------+
| 39|        State-gov| 77516.0|    Bachelors|         13.0|       Never-married|      Adm-clerical| Not-in-family|              White|   Male|      2174.0|         0.0|          40.0| United-States| <=50K|
| 50| Self-emp-not-inc| 83311.0|    Bachelors|         13.0|  Married-civ-spouse|   Exec-managerial|       Husband|              White|   Male|         0.0|         0.0|          13.0| United-States| <=50K|
| 38|          Private|215646.0|      HS-grad|          9.0|            Divorced| Handlers-cleaners| Not-in-family|              White|   Male|         0.0|         0.0|          40.0| United-States| <=50K|
| 53|          Private|234721.0|         11th|          7.0|  Married-civ-spouse| Handlers-cleaners|       Husband|              Black|   Male|         0.0|         0.0|          40.0| United-States| <=50K|
| 28|          Private|338409.0|    Bachelors|         13.0|  Married-civ-spouse|    Prof-specialty|          Wife|              Black| Female|         0.0|         0.0|          40.0|          Cuba| <=50K|
| 37|          Private|284582.0|      Masters|         14.0|  Married-civ-spouse|   Exec-managerial|          Wife|              White| Female|         0.0|         0.0|          40.0| United-States| <=50K|
| 49|          Private|160187.0|          9th|          5.0| Married-spouse-a...|     Other-service| Not-in-family|              Black| Female|         0.0|         0.0|          16.0|       Jamaica| <=50K|
| 52| Self-emp-not-inc|209642.0|      HS-grad|          9.0|  Married-civ-spouse|   Exec-managerial|       Husband|              White|   Male|         0.0|         0.0|          45.0| United-States|  >50K|
| 31|          Private| 45781.0|      Masters|         14.0|       Never-married|    Prof-specialty| Not-in-family|              White| Female|     14084.0|         0.0|          50.0| United-States|  >50K|
| 42|          Private|159449.0|    Bachelors|         13.0|  Married-civ-spouse|   Exec-managerial|       Husband|              White|   Male|      5178.0|         0.0|          40.0| United-States|  >50K|
| 37|          Private|280464.0| Some-college|         10.0|  Married-civ-spouse|   Exec-managerial|       Husband|              Black|   Male|         0.0|         0.0|          80.0| United-States|  >50K|
| 30|        State-gov|141297.0|    Bachelors|         13.0|  Married-civ-spouse|    Prof-specialty|       Husband| Asian-Pac-Islander|   Male|         0.0|         0.0|          40.0|         India|  >50K|
| 23|          Private|122272.0|    Bachelors|         13.0|       Never-married|      Adm-clerical|     Own-child|              White| Female|         0.0|         0.0|          30.0| United-States| <=50K|
| 32|          Private|205019.0|   Assoc-acdm|         12.0|       Never-married|             Sales| Not-in-family|              Black|   Male|         0.0|         0.0|          50.0| United-States| <=50K|
| 40|          Private|121772.0|    Assoc-voc|         11.0|  Married-civ-spouse|      Craft-repair|       Husband| Asian-Pac-Islander|   Male|         0.0|         0.0|          40.0|             ?|  >50K|
| 34|          Private|245487.0|      7th-8th|          4.0|  Married-civ-spouse|  Transport-moving|       Husband| Amer-Indian-Eskimo|   Male|         0.0|         0.0|          45.0|        Mexico| <=50K|
| 25| Self-emp-not-inc|176756.0|      HS-grad|          9.0|       Never-married|   Farming-fishing|     Own-child|              White|   Male|         0.0|         0.0|          35.0| United-States| <=50K|
| 32|          Private|186824.0|      HS-grad|          9.0|       Never-married| Machine-op-inspct|     Unmarried|              White|   Male|         0.0|         0.0|          40.0| United-States| <=50K|
| 38|          Private| 28887.0|         11th|          7.0|  Married-civ-spouse|             Sales|       Husband|              White|   Male|         0.0|         0.0|          50.0| United-States| <=50K|
| 43| Self-emp-not-inc|292175.0|      Masters|         14.0|            Divorced|   Exec-managerial|     Unmarried|              White| Female|         0.0|         0.0|          45.0| United-States|  >50K|
+---+-----------------+--------+-------------+-------------+--------------------+------------------+--------------+-------------------+-------+------------+------------+--------------+--------------+------+
```

The original size of the train data in the dataframe 32,561 rows and 15 columns including the label column. Below are the descriptions of each column. There is a mix of integer, string (categorical), and double variables. The columns must be converted to double or integer to satisfy the model input data requirements.



```
Dataframe size (32561, 15)
root
 |-- age: integer (nullable = true)
 |-- workclass: string (nullable = true)
 |-- fnlwgt: double (nullable = true)
 |-- education: string (nullable = true)
 |-- education-num: double (nullable = true)
 |-- marital-status: string (nullable = true)
 |-- occupation: string (nullable = true)
 |-- relationship: string (nullable = true)
 |-- race: string (nullable = true)
 |-- sex: string (nullable = true)
 |-- capital-gain: double (nullable = true)
 |-- capital-loss: double (nullable = true)
 |-- hours-per-week: double (nullable = true)
 |-- native-country: string (nullable = true)
 |-- label: string (nullable = true)
```

Data processing steps for both train and test data:

- Remove missing values: the data contains "?" that represent missing values. Remove the rows containing question marks.
- Drop columns "***fnlwgt***", "***education***" that are either not useful or duplicated
- Convert column "***label***" into 1 if ">50K"and 0 if "<=50K"
- Convert numerical columns to integer type: ***age, education-num, capital-gain, capital-loss, hours-per-week***
- Encode the categorical columns: ***workclass, marital_status, occupation, relationship, race, sex, country_encoded***

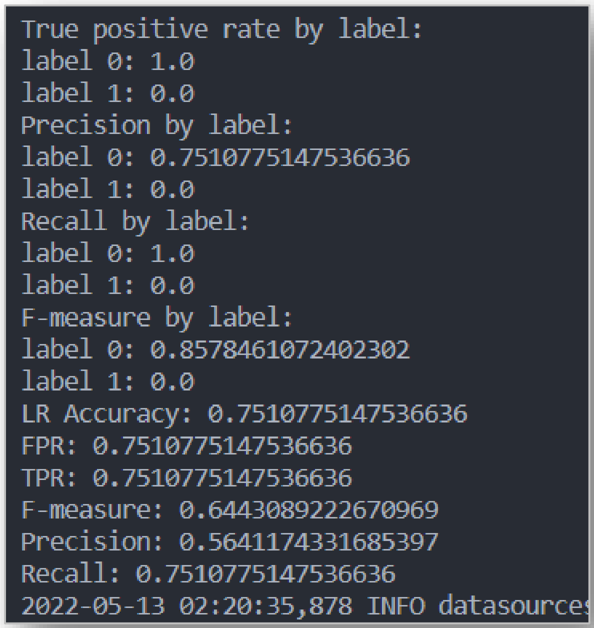
- Create feature vector from the feature columns: ***'age', 'workclass_encoded', 'education-num', 'marital_status_encoded', 'occupation_encoded', 'relationship_encoded', 'race_encoded', 'sex_encoded', 'capital-gain', 'capital-loss', 'hours-per-week', 'country_encoded'***

Model and Results:

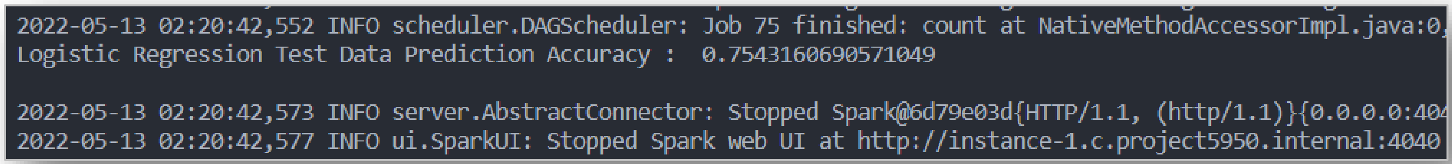
The model used is Binary LogisticRegression from the pyspark.ml.classification library.

The accuracy is 75.43%

Below is the snapshot of the model performance results:

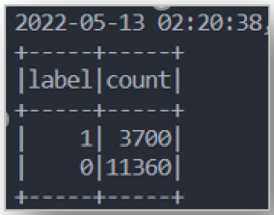


```
True positive rate by label:
label 0: 1.0
label 1: 0.0
Precision by label:
label 0: 0.7510775147536636
label 1: 0.0
Recall by label:
label 0: 1.0
label 1: 0.0
F-measure by label:
label 0: 0.8578461072402302
label 1: 0.0
LR Accuracy: 0.7510775147536636
FPR: 0.7510775147536636
TPR: 0.7510775147536636
F-measure: 0.6443089222670969
Precision: 0.5641174331685397
Recall: 0.7510775147536636
2022-05-13 02:20:35,878 INFO datasources
```



```
2022-05-13 02:20:42,552 INFO scheduler.DAGScheduler: Job 75 finished: count at NativeMethodAccessorImpl.java:0,
Logistic Regression Test Data Prediction Accuracy :  0.7543160690571049

2022-05-13 02:20:42,573 INFO server.AbstractConnector: Stopped Spark@6d79e03d{HTTP/1.1, (http/1.1)}{0.0.0.0:404
2022-05-13 02:20:42,577 INFO ui.SparkUI: Stopped Spark web UI at http://instance-1.c.project5950.internal:4040
```

Distribution of predicted classes:



```
2022-05-13 02:20:38,
+-----+-----+
|label|count|
+-----+-----+
|    1| 3700|
|    0|11360|
+-----+-----+
```

Ways to improve the prediction accuracy:

- Evaluate the feature importance and remove the ones that do not influence the class assignment.
- Bin the values with too many categories to create fewer categories. The columns ***age, capital gain, capital loss, hours per week, native country*** are the columns with more than 20 distinct categories each.

P4. Decision Tree and Random Forest Classifiers on Census Income Data

Overview

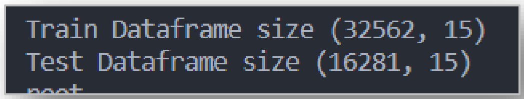
This part is going to use Decision Tree and Random Forest Classifiers in Spark ML to process the Census Income Data and predict the income level based of the features in dataset.

Dataset

The dataset is Census Dataset from the UCI Machine Learning Repository (link). The same dataset used in Part 3. The data description and data preprocessing are the same as in Part 3.

Model and Results:

Both Decision Tree and Random Forest models use the same train and test datasets. The train and test datasets sizes are:

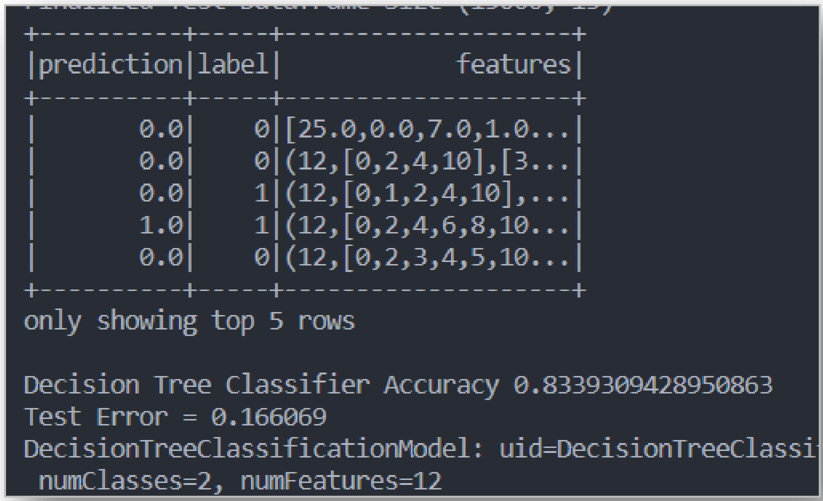


```
Train Dataframe size (32562, 15)
Test Dataframe size (16281, 15)
```

a) Decision Tree

The model used is DecisionTreeClassifier from pyspark.ml.classification library.

Decision Tree Classifier prediction accuracy is 83.39%

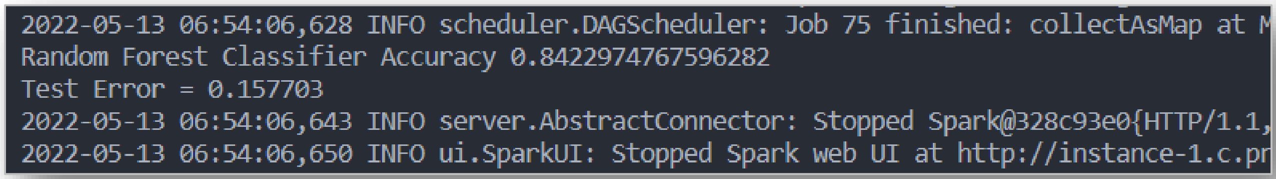


```
+----------+-----+--------------------+
|prediction|label|            features|
+----------+-----+--------------------+
|       0.0|    0|[25.0,0.0,7.0,1.0...|
|       0.0|    0|(12,[0,2,4,10],[3...|
|       0.0|    1|(12,[0,1,2,4,10],...|
|       1.0|    1|(12,[0,2,4,6,8,10...|
|       0.0|    0|(12,[0,2,3,4,5,10...|
+----------+-----+--------------------+
only showing top 5 rows

Decision Tree Classifier Accuracy 0.8339309428950863
Test Error = 0.166069
DecisionTreeClassificationModel: uid=DecisionTreeClassi
 numClasses=2, numFeatures=12
```

b) Random Forest

The model used is RandomForestClassifier rom pyspark.ml.classification library.

Random Forest Classifier prediction accuracy is 84.23%



```
2022-05-13 06:54:06,628 INFO scheduler.DAGScheduler: Job 75 finished: collectAsMap at M
Random Forest Classifier Accuracy 0.8422974767596282
Test Error = 0.157703
2022-05-13 06:54:06,643 INFO server.AbstractConnector: Stopped Spark@328c93e0{HTTP/1.1,
2022-05-13 06:54:06,650 INFO ui.SparkUI: Stopped Spark web UI at http://instance-1.c.pr
```

The columns ***age, capital gain, capital loss, hours per week, native country*** each have more than 20 distinct categories which negatively affects performance of both Decision tree and Random Forest classifier models. To improve the performance of these classifier models the data columns with too many categories need to be binned to create fewer distinct values.