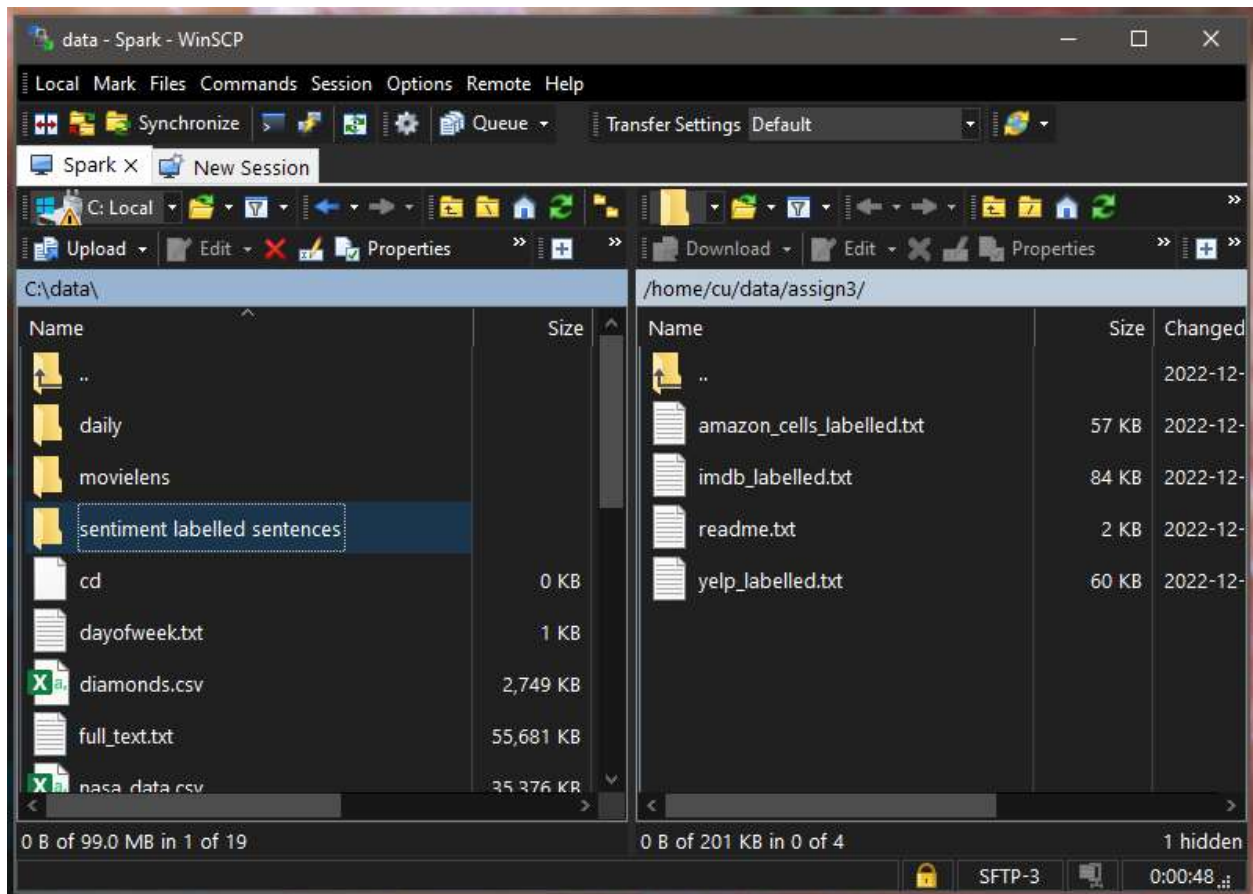


**Question 1:**

Using WinSCP, the 'Sentiment Labelled Sentences' folder was transferred into the /home/cu/data folder and renamed as assign3 folder to store the data files:

**Load the data file as Spark DataFrame.**

```
>>> df_imdb = spark.read
    .format('csv')
    .option('delimiter', '\t')
    .schema('sentence string, score integer')
    .load('/home/cu/data/assign3/imdb_labelled.txt')
```

```
cu@spark3cuvvm: ~
>>> df_imdb = spark.read.format('csv').option('delimiter', '\t').schema('sentence string, score integer').load('/home/cu/data/assign3/imdb_labelled.txt')
>>> df_imdb.show(20)
+-----+-----+
|          sentence|score|
+-----+-----+
|A very, very, ver...|    0|
|Not sure who was ...|    0|
|Attempting artine...|    0|
|Very little music...|    0|
|The best scene in...|    1|
|The rest of the m...|    0|
|Wasted two hours.  |    0|
|Saw the movie tod...|    1|
|A bit predictable. |    0|
|Loved the casting...|    1|
|And those baby ow...|    1|
|The movie showed ...|    1|
|The Songs Were Th...|    1|
|It Was So Cool.   |    1|
|This is a very "r...|    1|
|It had some avera...|    0|
|This review is lo...|    1|
|I'll put this gem...|    1|
|It's practically ...|    1|
|The structure of...| null|
+-----+-----+
only showing top 20 rows

>>> df_imdb.count()
1000
>>>
```

We can see that there are null values within the dataset and to prepare the dataset for processing we can remove them (data cleaning). There were only 5 null values out of 1000 records therefore the omission of the null values will not affect the dataset.

```
>>> df_imdb = df_imdb.na.drop()
>>> df_imdb.count()
```

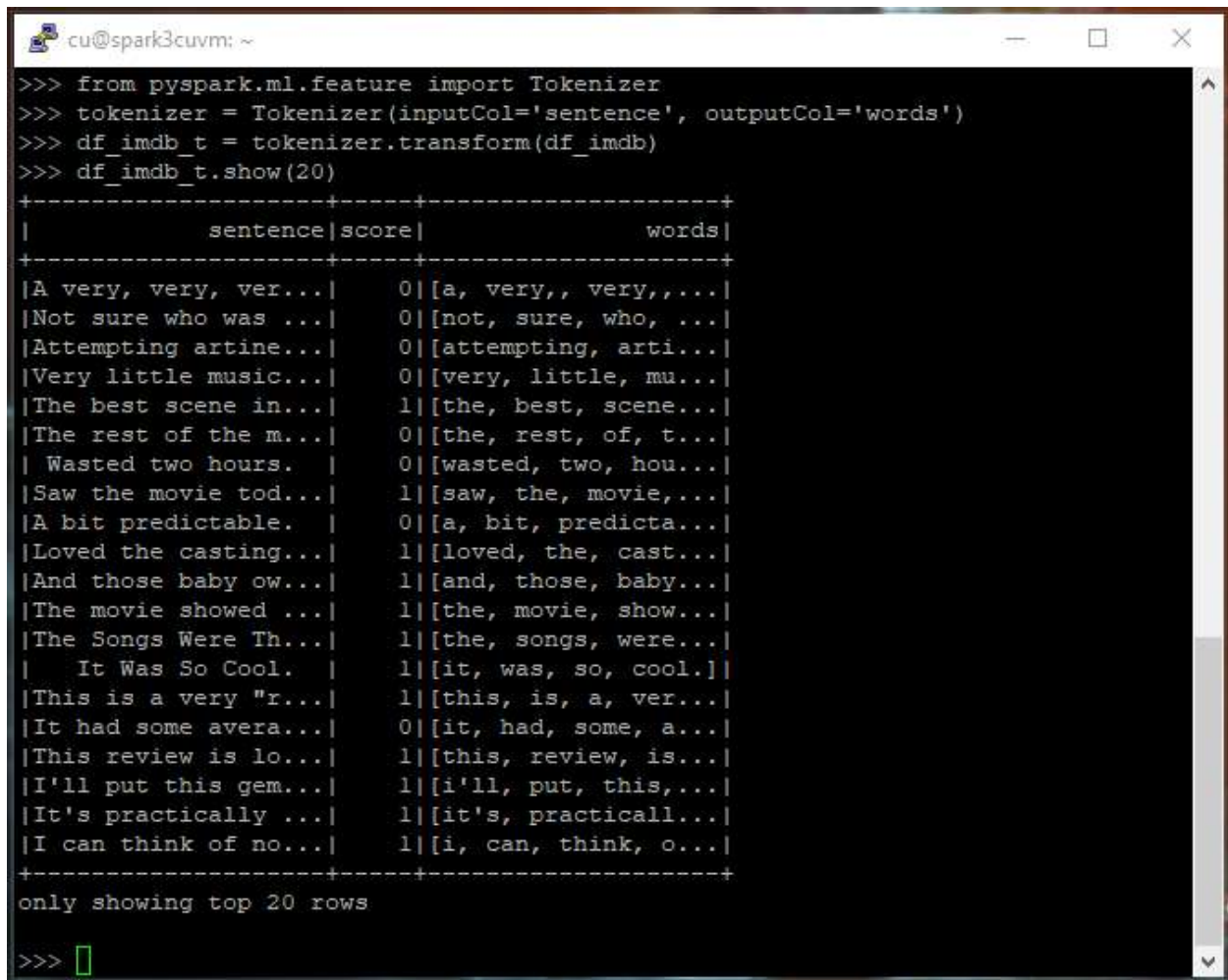
```
cu@spark3cuvvm: ~
>>> df_imdb.count()
1000
>>> df_imdb = df_imdb.na.drop()
>>> df_imdb.count()
995
>>>
```

## **Question 2:**

**Tokenize the reviews text into words.**

```
>>> from pyspark.ml.feature import Tokenizer
```

```
>>> tokenizer = Tokenizer(inputCol='sentence', outputCol='words')
>>> df_imdb_t = tokenizer.transform(df_imdb)
>>> df_imdb_t.show(20)
```



```
cu@spark3cuvvm: ~
>>> from pyspark.ml.feature import Tokenizer
>>> tokenizer = Tokenizer(inputCol='sentence', outputCol='words')
>>> df_imdb_t = tokenizer.transform(df_imdb)
>>> df_imdb_t.show(20)
+-----+-----+-----+
|      sentence|score|      words|
+-----+-----+-----+
|A very, very, ver...|    0|[a, very,, very,...|
|Not sure who was ...|    0|[not, sure, who, ...|
|Attempting artine...|    0|[attempting, arti...|
|Very little music...|    0|[very, little, mu...|
|The best scene in...|    1|[the, best, scene...|
|The rest of the m...|    0|[the, rest, of, t...|
| Wasted two hours. |    0|[wasted, two, hou...|
|Saw the movie tod...|    1|[saw, the, movie,...|
|A bit predictable. |    0|[a, bit, predicta...|
|Loved the casting...|    1|[loved, the, cast...|
|And those baby ow...|    1|[and, those, baby...|
|The movie showed ...|    1|[the, movie, show...|
|The Songs Were Th...|    1|[the, songs, were...|
| It Was So Cool.  |    1|[it, was, so, cool.]|
|This is a very "r...|    1|[this, is, a, ver...|
|It had some avera...|    0|[it, had, some, a...|
|This review is lo...|    1|[this, review, is...|
|I'll put this gem...|    1|[i'll, put, this,...|
|It's practically ...|    1|[it's, practicall...|
|I can think of no...|    1|[i, can, think, o...|
+-----+-----+-----+
only showing top 20 rows

>>> █
```

### Question 3:

Transform the reviews text data into numeric features using HashingTF class. Report how many features are created.

```
>>> from pyspark.ml.feature import HashingTF
>>> hashingTF = HashingTF(inputCol='words', outputCol='features')
>>> df_imdb_h = hashingTF.transform(df_imdb_t)
>>> df_imdb_h.show(20)
```

```

cu@spark3cuvms: ~
>>> from pyspark.ml.feature import HashingTF
>>> hashingTF = HashingTF(inputCol='words', outputCol='features')
>>> df_imdb_h = hashingTF.transform(df_imdb_t)
>>> df_imdb_h.show(20)
+-----+-----+-----+-----+
| sentence|score| words| features|
+-----+-----+-----+-----+
|A very, very, ver...| 0|[a, very,, very,,...|(262144,[18700,30...|
|Not sure who was ...| 0|[not, sure, who, ...|(262144,[1619,603...|
|Attempting artine...| 0|[attempting, arti...|(262144,[8039,322...|
|Very little music...| 0|[very, little, mu...|(262144,[27576,39...|
|The best scene in...| 1|[the, best, scene...|(262144,[17734,24...|
|The rest of the m...| 0|[the, rest, of, t...|(262144,[3924,187...|
| Wasted two hours. | 0|[wasted, two, hou...|(262144,[11275,54...|
|Saw the movie tod...| 1|[saw, the, movie,...|(262144,[23769,30...|
|A bit predictable. | 0|[a, bit, predicta...|(262144,[104421,1...|
|Loved the casting...| 1|[loved, the, cast...|(262144,[7014,958...|
|And those baby ow...| 1|[and, those, baby...|(262144,[102929,1...|
|The movie showed ...| 1|[the, movie, show...|(262144,[30950,92...|
|The Songs Were Th...| 1|[the, songs, were...|(262144,[6946,841...|
| It Was So Cool. | 1|[it, was, so, cool.](262144,[23685,30...|
|This is a very "r...| 1|[this, is, a, ver...|(262144,[48448,55...|
|It had some avera...| 0|[it, had, some, a...|(262144,[9781,192...|
|This review is lo...| 1|[this, review, is...|(262144,[17291,19...|
|I'll put this gem...| 1|[i'll, put, this,...|(262144,[2299,595...|
|It's practically ...| 1|[it's, practicall...|(262144,[34116,42...|
|I can think of no...| 1|[i, can, think, o...|(262144,[4967,153...|
+-----+-----+-----+-----+
only showing top 20 rows

>>> 

```

One feature vector is created for every record; therefore **995 feature vectors were created** (as our dataset contains 995 records after we filtered out the 5 records with null values). We can see that the vector length for the feature vectors is 262,144 in size (the default feature dimension length) followed by two other values within the vector, vector indices and vector values respectively.

#### Question 4:

Split the data into train (80%) and test (20%). For reproducibility, fix the seed as 11. Train a logistic regression model to classify the reviews in positive or negative category.

```
>>> df_train, df_test = df_imdb_h.randomSplit([0.8, 0.2], seed=11)
```



```

cu@spark3cuvvm: ~
>>> train, test = df_imdb.randomSplit([0.8, 0.2], seed=11)
>>> train.show(10)
+-----+-----+-----+-----+
| sentence|score| words| features|
+-----+-----+-----+-----+
| I love it. | 1| null|[, i, love, it., ...|(262144,[19036,92...|
| In fact, it's ha...| null|[, in, fact,, it'...|(262144,[2437,880...|
| The structure of...| null|[, the, structure...|(262144,[17291,87...|
| With great sound...| null|[, with, great, s...|(262144,[5561,190...|
| " But "Storm Troo...| 0|[" , but, "storm, ...|(262144,[14376,27...|
| (very serious spo...| 0|[(very, serious, ...|(262144,[45190,76...|
| ) Don't waste you...| 0|[, don't, waste,...|(262144,[7221,550...|
| ) What makes this...| 1|[, what, makes, ...|(262144,[7014,127...|
| ) a happy, wonder...| 1|[, a, happy,, wo...|(262144,[61899,88...|
| ) some great musi...| 1|[, some, great, ...|(262144,[14212,19...|
+-----+-----+-----+-----+
only showing top 10 rows

>>> test.show(10)
+-----+-----+-----+-----+
| sentence|score| words| features|
+-----+-----+-----+-----+
| & That movie was ...| 0|[, &, that, movie, ...|(262144,[48448,51...|
| (My mother and br...| 1|[(my, mother, and...|(262144,[3503,190...|
| ) very bad perfor...| 0|[, very, bad, pe...|(262144,[17893,42...|
| ), and sometimes ...| 0|[, , and, sometim...|(262144,[14570,19...|
| ***SPOILERS*** Wh...| 1|[, ***spoilers***, ...|(262144,[3665,877...|
| 1/10 - and only b...| 0|[, 1/10, -, and, on...|(262144,[25599,38...|
| 10 out of 10 for ...| 1|[, 10, out, of, 10,...|(262144,[1294,879...|
| 20th Century Fox'...| 0|[, 20th, century, f...|(262144,[27183,33...|
| A film not easily...| 1|[, a, film, not, ea...|(262144,[17291,25...|
| A good film by a ...| 1|[, a, good, film, b...|(262144,[17291,20...|
+-----+-----+-----+-----+
only showing top 10 rows

>>>

```

```

>>> from pyspark.ml.classification import LogisticRegression
>>> lr = LogisticRegression()
>>> lr.setLabelCol('score')
>>> lr_fit_model = lr.fit(df_train)
>>> model = lr_fit_model.transform(df_train)
>>> model.show(20)

```

```
cu@spark3cuvvm: ~
>>> from pyspark.ml.classification import LogisticRegression
>>> lr = LogisticRegression()
>>> lr.setLabelCol('score')
LogisticRegression_2cc73115cca7
>>> lr_fit_model = lr.fit(df_train)
>>> model = lr_fit_model.transform(df_train)
>>> model.show(20)
+-----+-----+-----+-----+-----+-----+
| sentence|score| words| features| rawPrediction| probability|prediction|
+-----+-----+-----+-----+-----+-----+
| " But "Storm Troo...| 0|[", but, "storm, ...|(262144,[14376,27...|[36.8207941403103...|[1.0,1.0207803130...| 0.0|
| & That movie was ...| 0|&, that, movie, ...|(262144,[48448,51...|[19.0252793169378...|[0.99999999453706...| 0.0|
| (My mother and br...| 1|[(my, mother, and...|(262144,[3503,190...|[-23.348766498150...|[7.24034983304410...| 1.0|
| (very serious spo...| 0|[(very, serious, ...|(262144,[45190,76...|[19.9130710548786...|[0.99999999775165...| 0.0|
| ) Don't waste you...| 0|[], don't, waste,...|(262144,[7221,550...|[19.0777285914495...|[0.99999999481620...| 0.0|
| ) some great musi...| 1|[], some, great, ...|(262144,[14212,19...|[-20.883952214442...|[8.51559096603388...| 1.0|
| ) very bad perfor...| 0|[], very, bad, pe...|(262144,[17893,42...|[20.1260574170332...|[0.99999999818296...| 0.0|
| ) , and sometimes ...| 0|[], , and, sometim...|(262144,[14570,19...|[19.1907974437889...|[0.99999999537040...| 0.0|
| ***SPOILERS*** Wh...| 1|***spoilers***, ...|(262144,[3665,877...|[-21.193504055463...|[6.24853398511381...| 1.0|
| ) 1 hour 54, min...| 0|[l, hour, 54, min...|(262144,[6767,221...|[21.3540259168680...|[0.99999999946781...| 0.0|
| ) 10/10 | 1| [10/10]|(262144,[137214],...|[-18.633893650099...|[8.07984087593737...| 1.0|
| :) Anyway, the pl...| 1|[:), anyway,, the...|(262144,[40698,46...|[-21.039308877100...|[7.29028071480537...| 1.0|
| :) Recommend with...| 1|[:), recommend, w...|(262144,[68228,89...|[-20.795736453360...|[9.30093074863485...| 1.0|
| A Lassie movie wh...| 0|[a, lassie, movie...|(262144,[27576,33...|[20.0502723556014...|[0.99999999803990...| 0.0|
| A cheap and cheer...| 0|[a, cheap, and, c...|(262144,[1010,111...|[26.5847516998852...|[0.9999999999715...| 0.0|
| A film not easily...| 1|[a, film, not, ea...|(262144,[17291,25...|[-19.165097665694...|[4.75011185647837...| 1.0|
| A good commentary...| 1|[a, good, comment...|(262144,[17291,51...|[-21.649489759473...|[3.96046392060024...| 1.0|
| A good film by a ...| 1|[a, good, film, b...|(262144,[17291,20...|[-25.246912806964...|[1.08493840517745...| 1.0|
| A great film by a...| 1|[a, great, film, ...|(262144,[17291,10...|[-38.234511824620...|[2.48291868246720...| 1.0|
| A piece of cinema...| 0|[a, piece, of, ci...|(262144,[44141,67...|[20.1452909818435...|[0.99999999821757...| 0.0|
+-----+-----+-----+-----+-----+-----+
only showing top 20 rows

>>>
```

**Question 5:**

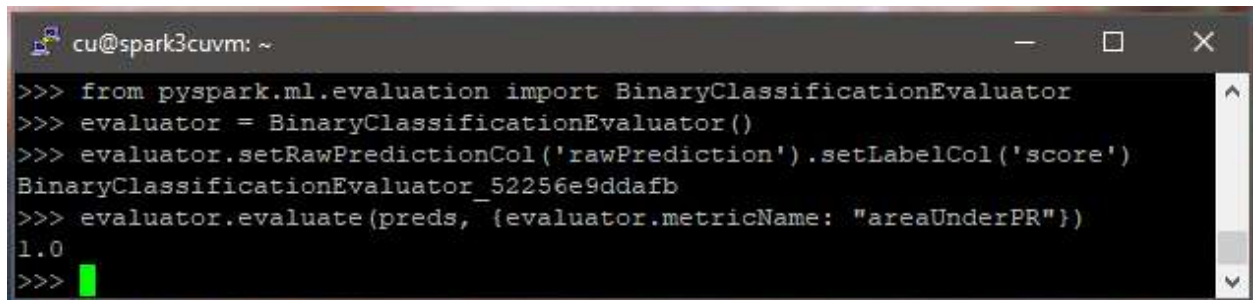
Apply test data to the trained LogisticRegression model trained in previous step. Compare the predicted values with the actual labels in terms of areaUnderROC using BinaryClassificationEvaluator class.

```
>>> lr_fit_model_preds = lr.fit(df_test)
>>> preds = lr_fit_model_preds.transform(df_test)
>>> preds.show(20)
```

```
cu@spark3cuvvm: ~
>>> lr_fit_model_preds = lr.fit(df_test)
>>> preds = lr_fit_model_preds.transform(df_test)
>>> preds.show(20)
+-----+-----+-----+-----+-----+-----+
| sentence|score| words| features| rawPrediction| probability|prediction|
+-----+-----+-----+-----+-----+-----+
| ) What makes this...| 1|[], what, makes, ...|(262144,[7014,127...|[-21.390315187170...|[5.13220680068921...| 1.0|
| ) a happy, wonder...| 1|[], a, happy,, wo...|(262144,[61899,88...|[-20.181620812337...|[1.71883205587003...| 1.0|
| ) 1/10 - and only b...| 0|[l/10, -, and, on...|(262144,[25599,38...|[19.7761848294627...|[0.99999999742182...| 0.0|
| ) 10 out of 10 for ...| 1|[10, out, of, 10,...|(262144,[1294,879...|[-19.500093293649...|[3.39795078592902...| 1.0|
| ) 10 out of 10 star...| 1|[10, out, of, 10,...|(262144,[87985,18...|[-19.250129242098...|[4.36289832733246...| 1.0|
| ) 10/10 | 1| [10/10]|(262144,[137214],...|[-18.677592103039...|[7.73436762886034...| 1.0|
| ) 20th Century Fox'...| 0|[20th, century, f...|(262144,[27183,33...|[21.0573322560976...|[0.99999999928399...| 0.0|
| ) A bit predictable...| 0|[a, bit, predicta...|(262144,[104421,1...|[19.2053452726482...|[0.99999999543727...| 0.0|
| ) A mature, subtle ...| 1|[a, mature,, subt...|(262144,[15494,15...|[-21.741999592037...|[3.61051830756534...| 1.0|
| ) A standout scene...| 1|[a, standout, sce...|(262144,[28492,62...|[-19.210803848696...|[4.53788926214464...| 1.0|
| ) A very good film ...| 1|[a, very, good, f...|(262144,[17291,18...|[-19.828901651079...|[2.44578022737290...| 1.0|
| ) About half way th...| 0|[about, half, way...|(262144,[17252,17...|[19.5033869778324...|[0.99999999661322...| 0.0|
| ) Actually, the gra...| 1|[actually,, the, ...|(262144,[95889,11...|[-19.373427541003...|[3.85680180489448...| 1.0|
| ) All in all, a gre...| 0|[all, in, all,, a...|(262144,[42404,68...|[19.0953362256061...|[0.99999999490668...| 0.0|
| ) All the pretty pi...| 0|[all, the, pretty...|(262144,[23071,34...|[20.0188975060436...|[0.99999999797743...| 0.0|
| ) And I forgot: The...| 1|[and, i, forgot:...|(262144,[7014,190...|[-21.149167017705...|[6.53180885344035...| 1.0|
| ) And his subtle co...| 1|[and, his, subtle...|(262144,[29440,49...|[-20.129631129500...|[1.81055731302943...| 1.0|
| ) And the rest of i...| 0|[and, the, rest, ...|(262144,[3924,694...|[-21.1560425600657...|[0.99999999935129...| 0.0|
| ) And, FINALLY, aft...| 0|[and,, finally,, ...|(262144,[2977,942...|[20.8337377483891...|[0.99999999910458...| 0.0|
| ) Aside from it's t...| 0|[aside, from, it'...|(262144,[2393,172...|[20.0317728863196...|[0.99999999800330...| 0.0|
+-----+-----+-----+-----+-----+-----+
only showing top 20 rows

>>>
```

```
>>> from pyspark.ml.evaluation import BinaryClassificationEvaluator
>>> evaluator = BinaryClassificationEvaluator()
>>> evaluator.setRawPredictionCol('rawPrediction').setLabelCol('score')
>>> evaluator.evaluate(preds, {evaluator.metricName: "areaUnderPR"})
```



```
cu@spark3cuvrn: ~
>>> from pyspark.ml.evaluation import BinaryClassificationEvaluator
>>> evaluator = BinaryClassificationEvaluator()
>>> evaluator.setRawPredictionCol('rawPrediction').setLabelCol('score')
BinaryClassificationEvaluator_52256e9ddafb
>>> evaluator.evaluate(preds, {evaluator.metricName: "areaUnderPR"})
1.0
>>>
```

Area under the ROC Curve value of the predicted values with the actual labels is 1.0, therefore we can say that this maximum value corresponds to a perfect classifier (error rate = 0).