>>> from pyspark.sql.types import \*

>>> from pyspark.sql.functions import \*

>>>

>>> from pyspark.ml import Pipeline

>>> from pyspark.ml.classification import LogisticRegression

>>> from pyspark.ml.feature import HashingTF, Tokenizer, StopWordsRemover

>>> from pyspark.sql import SQLContext

>>> sqlContext = SQLContext(sc)

>>> import sys

>>> ratings\_all = spark.read.csv('/user/mmishra2/review\_us.tsv',inferSchema=True, header=True,sep="\t")

19/05/19 04:59:31 WARN DataSource: Error while looking for metadata directory.

>>> ratings\_all.createOrReplaceTempView("table1")

>>> text = spark.sql(" select review\_headline, star\_rating from table1 where (review\_headline is not null and star\_rating is not null)")

>>> textdata = text.select("review\_headline", ((col("star\_rating") > 3).cast("Integer").alias("label")))

>>> textdata.show(truncate = False)

+--------------------------------------------------------+-----+

|review\_headline |label|

+--------------------------------------------------------+-----+

|ignore the review below |1 |

|awesome |1 |

|Read the book. It's good. |1 |

|Funniest book ever written about the Anti-Christ |1 |

|A winner that didn't last. Only a mothers love remained |1 |

|One of the best books I've ever read! |1 |

|Pulp Fiction Soundtrack: EXCELLENT!!!! |1 |

|The Doors: The best movie ever |1 |

|Gives a deeper understanding of chinese culture. |1 |

|One of the best marketing books ever written. Timeless.|1 |

|A definite page turner |1 |

|The best book that was ever written by Dean R. Koontz! |1 |

|Homeland is the best book ever!!!! |1 |

|Extraordinary story of courage through a child's eyes. |1 |

|How do you review a life? |1 |

|Kiss' return to hard rock from Disco. One of their best.|1 |

|The best book I've ever read. |1 |

|One of Stephen King's best books |1 |

|Infinite Jest: I've been waiting so long... |1 |

|Powerful work of Modern Philosophy; Compelling Story |1 |

+--------------------------------------------------------+-----+

only showing top 20 rows

>>> splits = textdata.randomSplit([0.7, 0.3],seed=0)

>>> textrain = splits[0]

>>> textest = splits[1].withColumnRenamed("label", "trueLabel")

>>> tokenizer = Tokenizer(inputCol="review\_headline", outputCol="Words")

>>> swr = StopWordsRemover(inputCol=tokenizer.getOutputCol(), outputCol="MeaningfulWords")

>>> hashTF = HashingTF(inputCol=swr.getOutputCol(), outputCol="features")

>>> lr = LogisticRegression(labelCol="label", featuresCol="features", maxIter=10, regParam=0.01)

>>> pipeline = Pipeline(stages=[tokenizer, swr, hashTF, lr])

>>> piplineModel = pipeline.fit(textrain)

>>> prediction = piplineModel.transform(textest)

>>> predicted = prediction.select("review\_headline", "prediction", "trueLabel")

>>> predicted.show(10)

+--------------------+----------+---------+

| review\_headline|prediction|trueLabel|

+--------------------+----------+---------+

| !| 1.0| 1|

| !| 1.0| 1|

| !| 1.0| 1|

| ! AMAZING !| 1.0| 1|

|! SHOGUN. To put...| 1.0| 1|

| !!!!!!!!!!!!!!!!| 1.0| 1|

|!!!!!!!!!Tron Eas...| 0.0| 1|

|!!!Soundgarden's ...| 1.0| 1|

|!!Better than SCR...| 1.0| 1|

|!!why do you not ...| 1.0| 1|

+--------------------+----------+---------+

only showing top 10 rows

>>> predicted10 = prediction.select("\*")

>>> predicted10.show(10)

+--------------------+---------+--------------------+--------------------+--------------------+--------------------+--------------------+----------+

| review\_headline|trueLabel| Words| MeaningfulWords| features| rawPrediction| probability|prediction|

+--------------------+---------+--------------------+--------------------+--------------------+--------------------+--------------------+----------+

| !| 1| [!]| [!]|(262144,[28990],[...|[-1.9165110009550...|[0.12825114044555...| 1.0|

| !| 1| [!]| [!]|(262144,[28990],[...|[-1.9165110009550...|[0.12825114044555...| 1.0|

| !| 1| [!]| [!]|(262144,[28990],[...|[-1.9165110009550...|[0.12825114044555...| 1.0|

| ! AMAZING !| 1| [!, , amazing, !]| [!, , amazing, !]|(262144,[16426,28...|[-4.1812429174841...|[0.01504955485364...| 1.0|

|! SHOGUN. To put...| 1|[!, , shogun., to...|[!, , shogun., pu...|(262144,[28990,75...|[-7.6556966211507...|[4.73116050743731...| 1.0|

| !!!!!!!!!!!!!!!!| 1| [!!!!!!!!!!!!!!!!]| [!!!!!!!!!!!!!!!!]|(262144,[17061],[...|[-2.4412566183861...|[0.08008029145348...| 1.0|

|!!!!!!!!!Tron Eas...| 1|[!!!!!!!!!tron, e...|[!!!!!!!!!tron, e...|(262144,[68210,10...|[0.11488381221507...|[0.52868940573382...| 0.0|

|!!!Soundgarden's ...| 1|[!!!soundgarden's...|[!!!soundgarden's...|(262144,[97484,22...|[-2.5280012860421...|[0.07391835095885...| 1.0|

|!!Better than SCR...| 1|[!!better, than, ...|[!!better, scream!!]|(262144,[161199,1...|[-4.4397296948707...|[0.01166153155627...| 1.0|

|!!why do you not ...| 1|[!!why, do, you, ...| [!!why, yet!!]|(262144,[38280,75...|[-2.6033679657968...|[0.06892197808279...| 1.0|

+--------------------+---------+--------------------+--------------------+--------------------+--------------------+--------------------+----------+

only showing top 10 rows

>>> tp = float(predicted.filter("prediction == 1.0 AND truelabel == 1").count())

>>> fp = float(predicted.filter("prediction == 1.0 AND truelabel == 0").count())

>>> tn = float(predicted.filter("prediction == 0.0 AND truelabel == 0").count())

>>> fn = float(predicted.filter("prediction == 0.0 AND truelabel == 1").count())

>>> metrics = spark.createDataFrame([

... ("TP", tp),

... ("FP", fp),

... ("TN", tn),

... ("FN", fn),

... ("Precision", tp / (tp + fp)),

... ("Recall", tp / (tp + fn))],["metric", "value"])

>>> metrics.show()

+---------+------------------+

| metric| value|

+---------+------------------+

| TP| 1651435.0|

| FP| 177198.0|

| TN| 189777.0|

| FN| 60527.0|

|Precision|0.9030981066184413|

| Recall|0.9646446591688367|

+---------+------------------+

>>> from pyspark.ml.evaluation import BinaryClassificationEvaluator

>>> evaluator = BinaryClassificationEvaluator(labelCol="trueLabel", rawPredictionCol="prediction", metricName="areaUnderROC")

>>> aur = evaluator.evaluate(prediction)

>>> print ("AUR = ", aur)

('AUR = ', 0.7408917144198975)

>>>