>>> from pyspark.ml.recommendation import ALS

>>> from pyspark.ml.evaluation import BinaryClassificationEvaluator, RegressionEvaluator

>>> from pyspark.ml.tuning import ParamGridBuilder, TrainValidationSplit, CrossValidator

>>> from pyspark.sql import SQLContext

>>> sqlContext = SQLContext(sc)

>>> import sys

>>>

>>> from pyspark.sql import functions as F

>>>

>>> from pyspark.ml import Pipeline

>>> from pyspark.ml.feature import VectorAssembler, StringIndexer, VectorIndexer, MinMaxScaler

>>> from pyspark.ml.tuning import CrossValidator, ParamGridBuilder, TrainValidationSplit

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

>>> ratings\_all.drop\_duplicates()

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

>>> df1=spark.sql("select distinct customer\_id, product\_category, star\_rating from table1")

>>> df1.show(5)

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

|customer\_id|product\_category|star\_rating|

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

| 52923248| Books| 5|

| 52867397| Books| 5|

| 52136665| Books| 5|

| 52855181| Books| 5|

| 52982798| Books| 5|

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

only showing top 5 rows

>>> df2=df1.filter(df1.star\_rating.isNotNull())

>>> new\_rating = df2

>>> strIdx = StringIndexer(inputCol = "product\_category", outputCol = "product\_categoryIdx")

>>> sm = strIdx.fit(new\_rating)

>>> new\_rating = sm.transform(new\_rating)

>>> new\_rating.show(5)

[Stage 9:======================================================> (27 + 1) / 28]19/05/19 03:48:51 WARN Executor: Managed memory leak detected; size = 5242880 bytes, TID = 515

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

|customer\_id|product\_category|star\_rating|product\_categoryIdx|

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

| 52923248| Books| 5| 4.0|

| 52867397| Books| 5| 4.0|

| 52136665| Books| 5| 4.0|

| 52855181| Books| 5| 4.0|

| 52982798| Books| 5| 4.0|

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

only showing top 5 rows

>>> data = new\_rating.select("customer\_id", "product\_categoryIdx", "star\_rating")

>>> splits = data.randomSplit([0.7, 0.3])

>>> train = splits[0].withColumnRenamed("star\_rating", "label")

>>> test = splits[1].withColumnRenamed("star\_rating", "trueLabel")

>>> train\_rows = train.count()

>>> test\_rows = test.count()

>>> print ("Training Rows:", train\_rows, " Testing Rows:", test\_rows)

('Training Rows:', 3767545, ' Testing Rows:', 1615203)

>>> als = ALS(userCol="customer\_id", itemCol="product\_categoryIdx", ratingCol="label")

>>> paramGrid = ParamGridBuilder() \

... .addGrid(als.rank, [1, 5]) \

... .addGrid(als.maxIter, [5, 10]) \

... .addGrid(als.regParam, [0.3, 0.1, 0.01]) \

... .addGrid(als.alpha, [2.0,3.0]) \

... .build()

>>> cv = TrainValidationSplit(estimator=als, evaluator=RegressionEvaluator(), estimatorParamMaps=paramGrid, trainRatio=0.8)

>>> model = cv.fit(train)

>>> prediction = model.transform(test)

>>> prediction = prediction.filter(prediction.prediction != float('nan'))

>>> prediction = prediction.withColumn("prediction", F.abs(F.round(prediction["prediction"],0)))

>>> evaluator = RegressionEvaluator(labelCol="trueLabel", predictionCol="prediction", metricName="rmse")

>>> rmse = evaluator.evaluate(prediction)

>>> print ("Root Mean Square Error (RMSE):", rmse)

('Root Mean Square Error (RMSE):', 2.105617149336906)

>>>