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

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

>>> from pyspark.sql import SparkSession

>>>

>>> from pyspark.ml import Pipeline

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

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

>>> from pyspark.ml.evaluation import BinaryClassificationEvaluator

>>>

>>> from pyspark.ml.regression import DecisionTreeRegressor

>>> from pyspark.ml.regression import GBTRegressor

>>> from pyspark.ml.evaluation import RegressionEvaluator

>>> from pyspark.ml.feature import OneHotEncoder

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

>>> from pyspark.ml.classification import LogisticRegression

>>> 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 20:25:48 WARN DataSource: Error while looking for metadata directory.

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

>>> tsv = spark.sql("SELECT \* FROM table1")

>>> tsv1 = tsv.select("product\_id", "product\_parent", "product\_title", "product\_category", "helpful\_votes", "total\_votes", "vine", "verified\_purchase", "review\_headline", "review\_body", "review\_date","star\_rating" )

>>> df1 = tsv1.filter(tsv1.product\_category.isNotNull())

>>> df2 = df1.filter(df1.helpful\_votes.isNotNull())

>>> df3 = df2.filter(df2.total\_votes.isNotNull())

>>> df4 = df3.filter(df3.vine.isNotNull())

>>> df5 = df4.filter(df4.verified\_purchase.isNotNull())

>>> df6 = df5.filter(df5.review\_headline.isNotNull())

>>> df7 = df6.filter(df6.review\_body.isNotNull())

>>> df8 = df7.filter(df7.review\_date.isNotNull())

>>> df9 = df8.filter(df8.star\_rating.isNotNull())

>>> data = df9.select("product\_id", "product\_parent", "product\_title", "product\_category", "helpful\_votes", "total\_votes", "vine", "verified\_purchase",col("star\_rating").alias("label"))

>>> data = StringIndexer(inputCol='product\_id', outputCol='product\_id'+"\_index").fit(data).transform(data)

>>> data = StringIndexer(inputCol='product\_title', outputCol='product\_title'+"\_index").fit(data).transform(data)

>>> data = StringIndexer(inputCol='product\_category', outputCol='product\_category'+"\_index").fit(data).transform(data)

>>> data = StringIndexer(inputCol='vine', outputCol='vine'+"\_index").fit(data).transform(data)

>>> data = StringIndexer(inputCol='verified\_purchase', outputCol='verified\_purchase'+"\_index").fit(data).transform(data)

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

>>> gbt\_train = splits[0]

>>> gbt\_test = splits[1].withColumnRenamed("label", "trueLabel")

>>> assembler = VectorAssembler(inputCols = ["product\_parent", "helpful\_votes", "total\_votes", "product\_id\_index", "product\_title\_index", "product\_category\_index", "vine\_index", "verified\_purchase\_index"], outputCol="features")

>>> gbt = GBTRegressor(featuresCol='features', labelCol='label', maxBins=90000, maxIter=1)

>>> gbt\_pipeline = Pipeline(stages=[assembler, gbt])

>>> paramGrid1 = (ParamGridBuilder()

... .addGrid(gbt.maxDepth,[2])

... .addGrid(gbt.minInfoGain,[0.0, 0.1])

... .addGrid(gbt.stepSize,[0.05, 0.1])

... .build())

>>> gbt\_tvs = TrainValidationSplit(estimator=gbt\_pipeline, evaluator=RegressionEvaluator(), estimatorParamMaps=paramGrid1, trainRatio=0.8)

>>> gbt\_model = gbt\_tvs.fit(gbt\_train)

>>> gbt\_prediction = gbt\_model.transform(gbt\_test)

>>> gbt\_predicted = gbt\_prediction.select("features", "prediction", "trueLabel")

>>> gbt\_predicted.show(10)

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

| features| prediction|trueLabel|

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

|[2.75786765E8,11....|3.8135135606421686| 5|

|[2.75786765E8,11....|3.8135135606421686| 5|

|[2.75786765E8,45....|3.8135135606421686| 1|

|[2.75786765E8,308...|3.8135135606421686| 5|

|[3.07844923E8,4.0...|3.8135135606421686| 5|

|[3.07844923E8,5.0...|3.8135135606421686| 4|

|[3.07844923E8,9.0...|3.8135135606421686| 5|

|[5.17434245E8,9.0...|3.8135135606421686| 5|

|[5.17434245E8,10....|3.8135135606421686| 5|

|[4.98635851E8,0.0...| 4.631383431440792| 5|

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

only showing top 10 rows

>>> gbt\_evaluator = RegressionEvaluator(labelCol="trueLabel", predictionCol="prediction", metricName="rmse")

>>> gbt\_rmse = gbt\_evaluator.evaluate(gbt\_prediction)

>>> print ("Root Mean Square Error (RMSE):", gbt\_rmse)

('Root Mean Square Error (RMSE):', 1.027869857781432