>>> 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 18:55:53 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])

>>> dt\_train = splits[0]

>>> dt\_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")

>>> dt = DecisionTreeRegressor(featuresCol='features', labelCol='label', maxBins=90000)

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

>>> paramGrid = ParamGridBuilder().build()

>>> cv = CrossValidator(estimator=dt\_pipeline, evaluator=RegressionEvaluator(), estimatorParamMaps=paramGrid, numFolds=2)

>>> dt\_model = cv.fit(dt\_train)

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

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

>>> dt\_predicted.show(10)

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

| features| prediction|trueLabel|

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

|[2.75786765E8,1.0...|4.9210522857349375| 5|

|[2.75786765E8,3.0...|4.9210522857349375| 5|

|[2.75786765E8,11....| 4.733121525019857| 5|

|[9.78255319E8,17....| 4.733121525019857| 5|

|[3.07844923E8,4.0...| 3.562660812501475| 5|

|[4.04458421E8,0.0...|4.9210522857349375| 4|

|[5.17434245E8,10....| 4.733121525019857| 5|

|[5.17434245E8,63....| 4.295602262176079| 5|

|[2.75786765E8,13....| 4.182404366306028| 5|

|[3.621552E7,3.0,4...|4.9210522857349375| 5|

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

only showing top 10 rows

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

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

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

('Root Mean Square Error (RMSE):', 0.9763653684438612)

>>>