##################################

Collaborative Filtering by User\_Song\_Playcounts

from pyspark.ml.evaluation import RegressionEvaluator

from pyspark.ml.recommendation import ALS

from pyspark.sql import SQLContext #####

from pyspark.sql import Row

from pyspark.sql.functions import \*

from pyspark.sql.functions import col

from pyspark.sql.types import IntegerType

lines = spark.read.text(r"C:\Users\kangy\Desktop\BigDataAnalytics\Project\Data\kaggle\_visible\_evaluation\_triplets.txt").rdd

sql\_c = SQLContext(sc)

uid = sql\_c.read.csv (r'C:\Users\kangy\Desktop\BigDataAnalytics\Project\Data\uid.csv')

sid = sql\_c.read.csv (r'C:\Users\kangy\Desktop\BigDataAnalytics\Project\Data\sid.csv')

parts = lines.map(lambda row: row.value.split("\t"))

ratingsRDD = parts.map(lambda p: Row(userId= p[0], songId= p[1],playcounts=float(p[2])))

ratings = spark.createDataFrame(ratingsRDD)

ratings = ratings.join(uid,uid.\_c0==ratings.userId)

ratings = ratings.select(col(“playcounts”).alias(“playcounts”), col(“songId”).alias(“songId”),col(“\_c1”).alias(“uid”))

ratings = ratings.join(sid,sid.\_c0==ratings.songId)

ratings = ratings.select(col(“playcounts”).alias(“playcounts”), col(“\_c1”).alias(“songId”),col(“uid”).alias(“userId”))

ratings = ratings.withColumn("songId", ratings["songId"].cast(IntegerType()))

ratings = ratings.withColumn("userId", ratings["userId"].cast(IntegerType()))

(training, test) = ratings.randomSplit([0.8, 0.2])

als = ALS(maxIter=5, regParam=0.01, userCol="userId", itemCol="songId", ratingCol="playcounts",coldStartStrategy="drop")

model = als.fit(training)

predictions = model.transform(test)

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

rmse = evaluator.evaluate(predictions)

print("Root-mean-square error = " + str(rmse))

userRecs = model.recommendForAllUsers(10)

userRecs = uid.join(userRecs,uid.\_c1 == userRecs.userId)

userRecs = userRecs.select(col(“\_c0”).alias(“userId”),col(“recommendations”).alias(“recommendations”))

userRecs.show() :top 10 recommendations for each user.

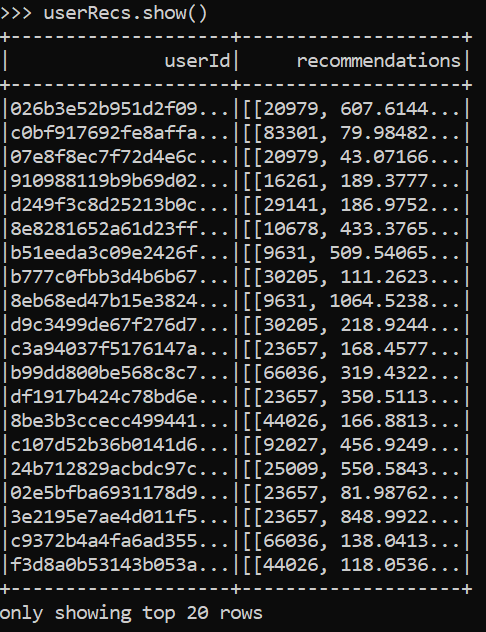
users = ratings.select(als.getUserCol()).distinct().limit(3)

userSubsetRecs = model.recommendForUserSubset(users, 10)

userSubsetRecs = userSubsetRecs.join(uid,uid.\_c1 == userSubsetRecs.userId)

userSubsetRecs = userSubsetRecs.select(col(“\_c0”).alias(“userId”),col(“recommendations”).alias(“recommendations”))

userSubsetRecs.show() top 10 recommendations for a specified set of users.



**##################################**

Popularity – Find Top 100 popular songs

rating = spark.createDataFrame(ratingsRDD)

pop = rating.groupby([“songId”]).agg({“playcounts”: “sum”})

pop.sort('sum(playcounts)',ascending=False).show(10)

