Cloud Data Storage
Project – Spark
Ishita Shah – 801034015

The Source code –

```
import os, sys
print "Current working dir : %s" % os.getcwd() --
Current working dir : /home/ubuntu/server/spark-2.3.0-bin-hadoop2.7

// gets the current directory that is
datasets_path = os.path.join('/home/ubuntu/server/documents/') // datapath , here is the data


small_ratings_doc (small_rating document) = os.path.join(datasets_path, 'ratings.csv')
SR_RWD  = sc.textFile(small_ratings_doc) // rating document
SR_RWD.count()
SR_RWD_header = SR_RWD.take(1)[0]
Sml_Ratingdata = SR_RWD.filter(lambda line: line!=SR_RWD_header).map(lambda line:
line.split(",")).map(lambda tokens: (tokens[0],tokens[1],tokens[2])).cache()
Sml_Ratingdata.take(5)
Small_Movie_doc = os.path.join(datasets_path,'movies.csv')
small_movies_raw_data = sc.textFile(Small_Movie_doc)
small_movies_raw_data.count()
small_movies_raw_data_header = small_movies_raw_data.take(1)[0]
small_movies_data = small_movies_raw_data.filter(lambda line:
line!=small_movies_raw_data_header).map(lambda line: line.split(",")).map(lambda tokens:
(tokens[0],tokens[1])).cache()
small_movies_data.take(8)

movie_TITLE = small_movies_data.map(lambda x: (int(x[0]),x[1]))
movie_TITLE.take(12)

TRAIN_RD, VALID_RD, small_test_RDD = Sml_Ratingdata.randomSplit([6, 2, 2], seed=0L)
VALIDATION_PREDICT_RD = VALID_RD.map(lambda x: (x[0], x[1]))
SML_TEST_PRED_RD = small_test_RDD.map(lambda x: (x[0], x[1]))

-----MATH Function----------
from pyspark.mllib.recommendation import ALS
import math

seed = 5L
iterations = 10
regularization_parameter = 0.1
ranks = [4, 8, 12,25,30,40,50]
errors = [0, 0, 0,0,0,0,0]
err = 0
tolerance = 0.02
```

```
min_error = float('inf')
best_rank = -1
best_iteration = -1


-- CALCULATION------
for rank in ranks:
   model = ALS.train(TRAIN_RD, rank, seed=seed,
iterations=iterations,lambda_=regularization_parameter)
   predictions = model.predictAll(VALIDATION_PREDICT_RD).map(lambda r: ((r[0], r[1]), r[2]))
   rates_and_preds = VALID_RD.map(lambda r: ((int(r[0]), int(r[1])), float(r[2]))).join(predictions)
   error = math.sqrt(rates_and_preds.map(lambda r: (r[1][0] - r[1][1])**2).mean())
   err += 1
          print 'For rank %s the RMSE is %s' % (rank, error)
   if error < min_error:
      min_error = error
      best_rank = rank
print 'The best model was trained with rank %s' % best_rank


(model.userFeatures().count())

model.productFeatures().count()

model.productFeatures().lookup(1020)[0]

import numpy as np
from numpy import linalg as LA

def cosineSimilarity(vec1, vec2):
  return vec1.dot(vec2) / (LA.norm(vec1) * LA.norm(vec2))

itemId = 1020

itemFactor = np.asarray(model.productFeatures().lookup(itemId))[0]

cosineSimilarity(itemFactor,itemFactor)

Pam = model.productFeatures().map(lambda
products:(products[0],cosineSimilarity(np.asarray(products[1]),
itemFactor))).join(movie_TITLE).map(lambda r: (r[1][1], r[1][0], r[0]))

Pam.take(15)

sortPam = Pam.takeOrdered(20, key=lambda x: -x[1])

sortPam
```