Import necessary spark and math dependencies

Initialize SparkContext and SparkSession

Read the ratings csv to an RDD

Remove the header from the RDD using the filter

For each line in the ratings rdd split on the commas and conver the userId, movieId and given rating to a pyspark recommendation Rating object

Set the rank and numIterations parameters for the Alternating Least Squares model

Train the ALS model on the processed ratings csv using the rank and numIterations parameters

**# Calculate RegressionMetrics**

#Create a test data set

For each line within ratings RDD map, remove the rating and map the user and movieId to the testData rdd

Use the ALS model to make predictions using the testData RDD.

Store the predictions in an RDD and then map the user and movie id tuple to the rating predicted

Create an RDD from the full ratings.csv mimicking the prediction RDD format, a tuple of user and movieId, and the movie rating

Join the ratings tuple RDD and the prediction RDD and select only the ratings tuple from the joined rdd to create a predictedVsActual rating RDD

Use the RegressionMetrics from the pyspark evaluation library on the predictedVsActual RDD

Use the RegressionMetrics library to calculate RootMeanSquaredError and MeanSquaredError

**# Calculate #RankingMetrics**

Map each user to a tuple of their corresponding rated movieIds and given rating, sort the tuple on the rating in descending order. Reduce by key on the user. Store the resulting RDD of user and tuple of (movieID and rating) in a rankedRatings rdd

Create a getMovie function that will split the rankedRatings line and return the userId and a list of the movie ratings

Create an rdd for the groundTruthRankedRatings and store the result of the getMovie function on the rankedRatings in this RDD

# Convert the predictions RDD into rankings

Group the predictions by key in case any duplicates were created (this was seldom seen)

Create a rankedTestRatings RDD and map the userId with a tuple of the movieId and then average the ResultIterable (incase there were duplicates within the prediction RDD). Sort on the rating in descending order, and then reduceByKey on the userID

Call the getMovie function on the rankedTestRatings RDD and store the result in the predictedRankedRatings RDD

Join the groundTruthRankedRatings with the predictedRankedRatings, creating an rdd of the userID and a list of the sorted predicted rated movie ids vs. the ground truth rated movie ids

Create a function convertToRankings that reads in the sorted movieID lists and converts the groundTruthRankings to a list of 1…n, store the index (1..n) for each movieId in a dictionary. Iterate through the predictedRankings and append to the list where the predicted movie ranking actually appeared in the groundTruthRanking

Create a rankingsRDD that runs the convertToRankings function on the joined ranked ratings RDD

Map the lists so that there is a rdd with the predictedRankingsList and the groundTruthRankingsList

Use pyspark evaluation to calculate the RankingMetrics and the mean average precision

**# Implement five fold cross validation and grid search**

Define the number of folds for cross validation

Split the ratings rdd in to k folds

Create a datalist that stores all k folds

# Implement grid search for cross validation

For rank in parameters

For numIterations in parameters

For the range in number of folds

Create the currentTesting rdd on the ith number from the dataList

Create the training rdd from the remaining folds in datalist

Union the training folds together into one RDD

Create the ALS model on the current training data and the current rank and numIterations parameters

Calculate the RegressionMetrics using the same methodology as above

Calculate the RankingMetrics using the same methodology as above

Output the RMSE, MSE and MAP along with the parameters used for the model