Import libraries that are required for data manipulation and pre-processing.

Reading the raw Dataset.

Analysis of Data.

Checking for Null values.

Mapping the loanID and AccountID then Saving the mapped Id’s.

Update the "Number\_Of\_Loans\_Granted\_\_c" column by decreasing the number by total number of loans row found with "Closed Won-Funded", “Closed Won-Payment Failed”, “Payment Plan”, “Debt Management” in the StageName column.

Now Removing the Rows containg the stagename as "Closed Won-Funded", “Closed Won-Payment Failed”, “Payment Plan”, “Debt Management”.

Creating the Ratings columns : Applying the Cumulative Data Option:sum of all the successful subtraction unsuccessful servicing of specific loan.

Formula used: No. of loans paid – (Number of loans granted – number of loans paid)

Applying the Scaling to this ratings column.

Formula used :

if rating<=10 and rating>=1:

return 1

elif rating>10 and rating<100:

return int(str(rating)[0])+1

elif rating>=100:

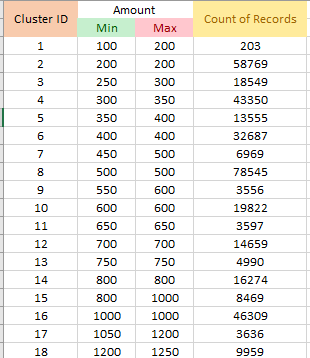
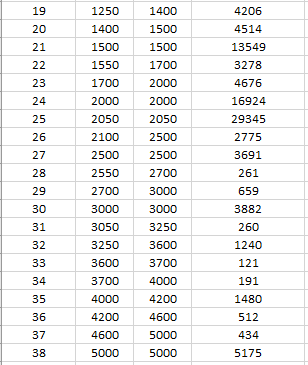
return 10

else:

return rating

so our ratings comes into an range of: {0, 1, 2, 3, 4, 5, 6, 7, 8}

Now performing the Clustering.

To Create the Loan ID.

Exporting the Dataframe.

Now Importing the Libraries from pySpark to perform the ALS, Cosine Similarity & Centered Cosine Similarity.

Creating the spark session

Reading/importing the Modified Data frame using spark read method.

Describing about the features

Aggregating the loans per user (getting all loans for each user user)

Converting the dataset into train & validation split.

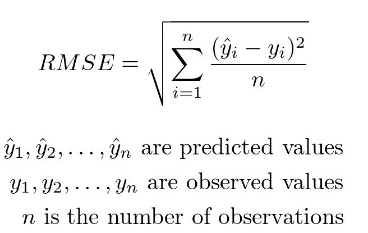
Performing the ALS matrix factorization using Grid Search Function to select the best model based on RMSE.

Defining the hyper parameters and Fitting the model on it.

Predicting the results using validation data.

## Error Evaluation with RMSE on Validation data

## using formula as :



Recommendation for N users of m loans (m=5) using ALS model .

User Factors and Item Factors to calculate the cosine Similarity.

Calculating the cosine similarity (User-Item Cosine Similarity)

Using formula as :

def cosine\_similarity(u1, u2):

"""Calculates cosine similarity between two user factors."""

# Convert lists to DenseVectors before performing operations

u1\_vector = Vectors.dense(u1)

u2\_vector = Vectors.dense(u2)

return u1\_vector.dot(u2\_vector) / (u1\_vector.norm(p=2) \* u2\_vector.norm(p=2))

Then Extracting the top m Recommendations.

Then Extracting the recommend top m loans not taken by the user

Now applying the MAP@k to evaluate the results of ALS Model & cosine similarity

Now Applying the Centered Cosine Similarity (User Item Centered Cosine Similarity)

Using formula as:

def centered\_cosine\_similarity(u, v):

""" Calculate centered cosine similarity between two vectors u and v. """

# Calculate mean of u and v

mean\_u = sum(u) / len(u)

mean\_v = sum(v) / len(v)

# Subtract mean from u and v

centered\_u = Vectors.dense([x - mean\_u for x in u])

centered\_v = Vectors.dense([x - mean\_v for x in v])

# Calculate centered cosine similarity

dot\_product = float(centered\_u.dot(centered\_v))

norm\_u = float(centered\_u.norm(2))

norm\_v = float(centered\_v.norm(2))

centered\_sim = dot\_product / (norm\_u \* norm\_v)

return centered\_sim

And applying the Centered Cosine Similarity.