In [ ]:
```
pip install pyspark
```

In [ ]:
```
from pyspark.sql import functions as f
from pyspark.sql import DataFrameNaFunctions as DFna
from pyspark.sql.functions import udf, col, when
import matplotlib.pyplot as plt
import pyspark as ps
import os, sys, requests, json
from pyspark.sql.functions import col,size,regexp_replace,lit
from pyspark.ml.evaluation import RegressionEvaluator
from pyspark.ml.recommendation import ALS

from pyspark.ml.evaluation import RegressionEvaluator
from pyspark.ml.recommendation import ALS
from pyspark.ml.tuning import CrossValidator, ParamGridBuilder
from pyspark.ml import Pipeline
from pyspark.sql import Row
import numpy as np
import math
from pyspark.sql.functions import regexp_replace
from pyspark.sql import SparkSession

spark = SparkSession.builder.master("local[*]").config("spark.executor.memory", "70g").config("spark.dr
iver.memory", "50g").config("spark.memory.offHeap.enabled",True).config("spark.memory.offHeap.size","20
g").appName("sampleCodeForReference").getOrCreate()

sc = spark.sparkContext
```

In [ ]:
```
from pyspark.sql import SQLContext
sqlContext = SQLContext(sc)
```

In [ ]:
```
pro=spark.read.csv('../input/data-science-for-good-careervillage/professionals.csv', header=True,quote=
'"',sep=",",multiLine=True)
email=spark.read.csv('../input/data-science-for-good-careervillage/emails.csv', header=True,quote='"',s
ep=",",multiLine=True)
ques=spark.read.csv('../input/data-science-for-good-careervillage/questions.csv', header=True,quote='"'
,sep=",",multiLine=True)
match=spark.read.csv('../input/data-science-for-good-careervillage/matches.csv', header=True,quote='"',
sep=",",multiLine=True)
ans=spark.read.csv('../input/data-science-for-good-careervillage/answers.csv', header=True,quote='"',se
p=",",multiLine=True)
ans_score=spark.read.csv('../input/data-science-for-good-careervillage/answer_scores.csv', header=True,
quote='"',sep=",",multiLine=True)
```

In [ ]:
```
pro.show(5)
```

In [ ]:
```
from pyspark.sql.functions import lit,row_number,col
from pyspark.sql.window import Window

w = Window().partitionBy(lit('a')).orderBy(lit('a'))

ques = ques.withColumn("ques_id", row_number().over(w))
ques_500=ques.filter(col("ques_id").between(1,20)).select('questions_id','ques_id')
```

In [ ]:
```
pro = pro.withColumn("pro_id", row_number().over(w))
pro_500=pro.filter(col("pro_id").between(1,10000)).select('professionals_id','pro_id')
```

In [ ]:
```
ans = ans.withColumn("row_num", row_number().over(w))
ans_300=ans.filter(col("row_num").between(1,20000))
```

In [ ]:
```
list_ques = np.array(ques_500.select('questions_id').collect())
```

In [ ]:
```
pro_ques=pro_500.withColumn("questions_id", f.lit(list_ques[0][0]))

for i in range(1,len(list_ques)):
    df=pro_500.withColumn("questions_id", f.lit(list_ques[i][0]))
    pro_ques = pro_ques.union(df)
```

In [ ]:
```
pro_ques_final=pro_ques.withColumn('combined',f.array(pro_ques.questions_id,pro_ques.professionals_id))
```

In [ ]:
```
ans_new=np.array(ans.withColumn('ans_combined',f.array(ans_300.answers_question_id,ans_300.answers_auth
or_id)).select('ans_combined').collect())
```

In [ ]:
```
ans_final=[]
for i in ans_new:
    ans_final.append(tuple(i[0]))
```

In [ ]:
```
from pyspark.sql.types import IntegerType
my_udf = udf(lambda pair: 1 if tuple(pair) in ans_final else 0, IntegerType())
pro_ques_final = pro_ques_final.withColumn('check', my_udf(pro_ques_final['combined'])).select('profess
ionals_id','questions_id','check')
```

In [ ]:
```
pro_ques_final=pro_ques_final.join(pro_500,pro_500.professionals_id==pro_ques_final.professionals_id).d
rop(pro_500.professionals_id).select('pro_id','professionals_id','questions_id','check')
```

In [ ]:
```
pro_ques_final=pro_ques_final.join(ques_500,ques_500.questions_id==pro_ques_final.questions_id).drop(qu
es_500.questions_id).select('pro_id','ques_id','check')
```

In [ ]:
```
pro_ques_final.toPandas().to_csv('collaborative_label0.csv')
```