a) Load Bank Marketing Dataset and create RDD

rdd1 = sc.textFile('/home/miles/futurense\_hadoop-pyspark/labs/dataset/bankmarket/bankmarketdata.csv')

b) Give marketing success rate. (No. of people subscribed / total no. of entries)

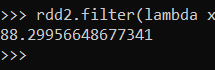
rdd2 = rdd1.map(lambda x:x.split(';')[16])

rdd2.filter(lambda x:x=='"yes"').count()/rdd2.count() \* 100



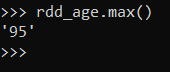
c) Give marketing failure rate

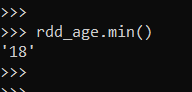
rdd2.filter(lambda x:x=='"no"').count()/rdd2.count() \* 100

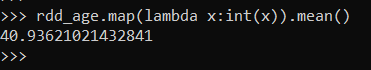


d) Maximum, Mean, and Minimum age of the average targeted customer

rdd\_age = rdd1.map(lambda x:x.split(';')[0]).filter(lambda x:x!='age')



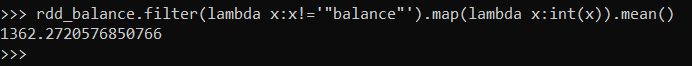




e) Check the quality of customers by checking the average balance, median balance of customers

rdd\_balance = rdd1.map(lambda x:x.split(';')[5])

Mean



Median

sorted\_rdd = rdd\_balance.filter(lambda x:x!='"balance"').sortBy(lambda x: x)

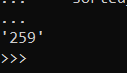
count = sorted\_rdd.count()

if count % 2 == 0:

sorted\_rdd.zipWithIndex().filter(lambda x:x[1] == (count/2 -1) or x[1] == count/2).map(lambda x:x[0]).mean()

else:

sorted\_rdd.zipWithIndex().filter(lambda x:x[1] == (count//2)).map(lambda x:x[0]).collect()[0]

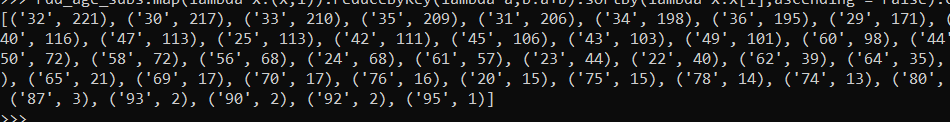


f) Check if age matters in marketing subscription for deposit

rdd3 = rdd1.map(lambda x:x.split(';'))

rdd\_age\_subs = rdd3.filter(lambda x:x[16]=='"yes"').map(lambda x:x[0])

rdd\_age\_subs.map(lambda x:(x,1)).reduceByKey(lambda a,b:a+b).sortBy(lambda x:x[1],ascending = False).collect()



g) Show AgeGroup [Teenagers, Youngsters, MiddleAgers, Seniors] wise Subscription Count.

age\_groups = {

"Teenagers": (13, 19),

"Youngsters": (20, 30),

"MiddleAgers": (31, 59),

"Seniors": (60, 130)

}

def categorize\_age(age):

for group, bounds in age\_groups.items():

if bounds[0] <= int(age) <= bounds[1]:

return group

return "”

rdd3.filter(lambda x: x[16]=='"yes"').map(lambda x:x[0]).map(lambda age: (categorize\_age(age), 1)).reduceByKey(lambda a, b: a + b).collect()



h) Check if marital status mattered for subscription to deposit.

rdd3.filter(lambda x:x[2] != '"marital"' and x[16]=='"yes"').map(lambda x:(x[2],1)).reduceByKey(lambda a,b:a+b).collect()



i) Check if age and marital status together mattered for subscription to deposit scheme

rdd3.filter(lambda x:x[2] != '"marital"' and x[0] != '"age"' and x[16]=='"yes"').map(lambda x:((x[0],x[2]),1)).reduceByKey(lambda a,b:a+b).sortBy(lambda x:x[1],ascending=False).collect()

