Q1 *Dataset: Covid19.txt*

Using the dataset present at the HDFS path

'hdfs:///bigdatads/common\_folder/assignment3/Covid19.txt', perform the following:

1. Write a program that converts all the words in the file to lowercase and splits every line of the document based on space
2. Remove the following objectionable words from the file: “Chinese-Virus”, “Wuhan-Virus” and “bio-weapon”
3. Count all the words from the RDD created in step 1 and produce an output in the format (word, count)

Output Format: Paste the commands used to solve this problem in the text box below. Also, attach a screenshot(s) of the code.

Ans1 cov\_loc=sc.textFile("/bigdatads/common\_folder/assignment3/covid19.txt.tmp")

covid=cov\_loc.collect()

low\_cov=[]

wor\_cov=[]

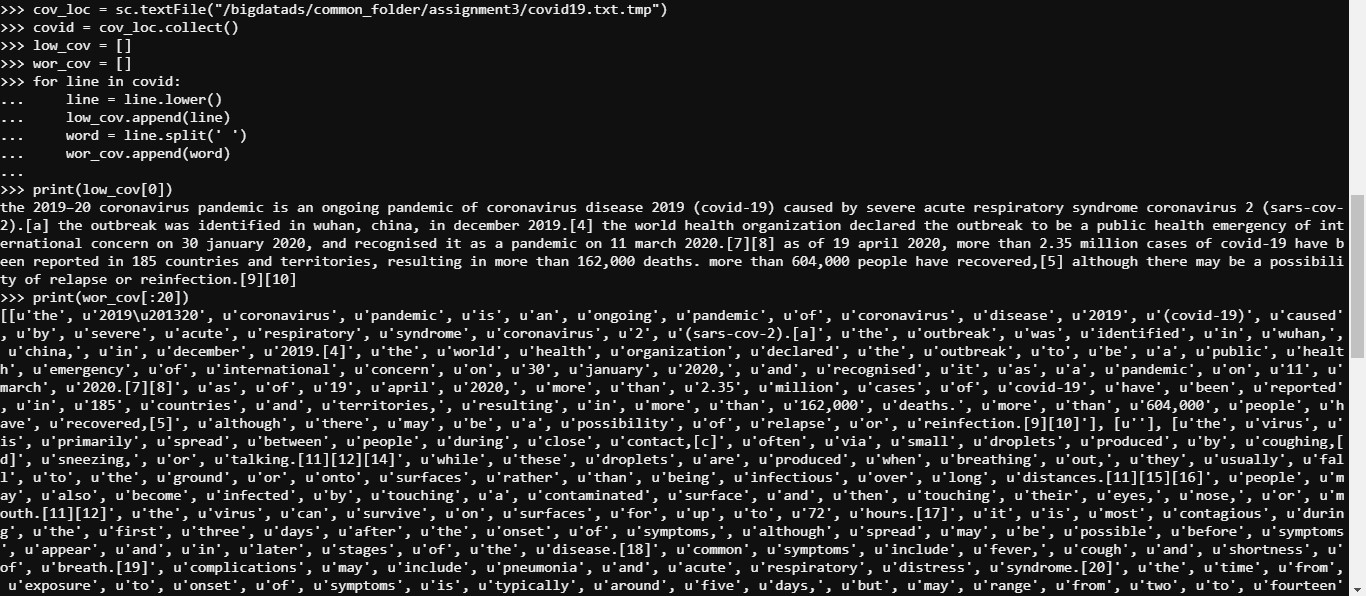
for line in covid:

... line=line.lower()

... low\_cov.append(line)

... word=line.split(' ')

... wor\_cov.append(word)



rem\_cov=[]

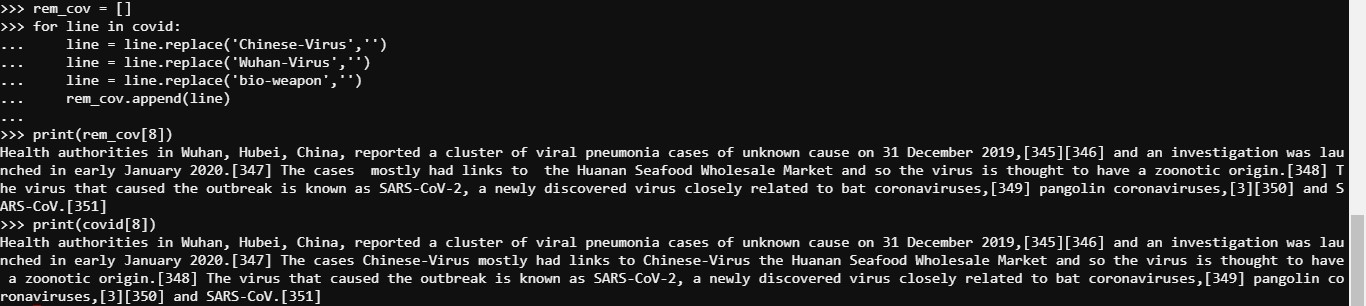
for line in covid:

... line=line.replace('Chinese-Virus','')

... line=line.replace('Wuhan-Virus','')

... line=line.replace('bio-weapon','')

... rem\_cov.append(line)



word\_list=[]

for i in range(len(wor\_cov)):

... for j in range(len(wor\_cov[i])):

... word\_list.append(wor\_cov[i][j])

word\_count=dict()

for word in word\_list:

... if word in word\_count:

... word\_count[word]+=1

... else:

... word\_count[word]=1



Q2 *Dataset: Covid19.txt*

Using the same given dataset, perform the following:

1. Load the dataset and create an RDD, perform sampling using half of the RDD data
2. On the partitioned RDD created in step 1, find out the number of occurrences of the following words: “pandemic” and “covid” in each of the partitions
3. From the RDD created in step 1, create one more sample with a quarter of the RDD data, then perform a union between the newly created RDD with the RDD created in step 1

Output Format: Paste the commands used to solve this problem in the text box below. Also, attach a screenshot(s) of the code.

Ans2 sp=cov\_loc.sample(False, 0.5, 1)

rt=str(sp.collect())

pan=0

cov=0

for line in rt:

... if("pandemic" in line):

... pan+=1

... elif("covid" in line):

... cov+=1

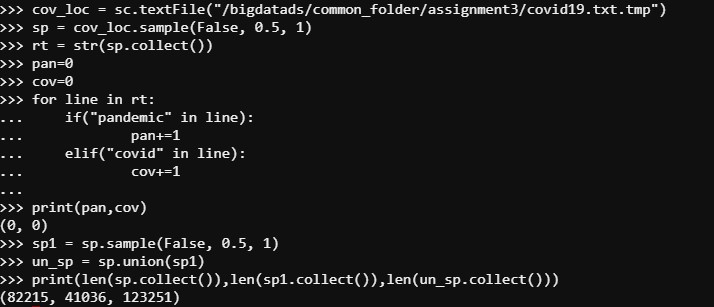
...

print(pan,cov)

sp1=sp.sample(False, 0.5, 1)

un\_sp=sp.union(sp1)

print(len(sp.collect()),len(sp1.collect()),len(un\_sp.collect()))



Q3 *Dataset: Covid19.txt*

Using the same given dataset, perform the following:

1. Load the dataset and create an RDD, get the distinct elements from that RDD and count the number of distinct elements
2. Find the number of partitions present in the RDD
3. Reduce the number of partitions present in the RDD to 1 and check whether the partitions got reduced

Output Format: Paste the commands used to solve this problem in the text box below. Also, attach a screenshot(s) of the code.

Ans3 cov\_loc=sc.textFile("/bigdatads/common\_folder/assignment3/covid19.txt.tmp")

co\_dis=cov\_loc.distinct()

dis\_ele=co\_dis.collect()

print(dis\_ele[1])

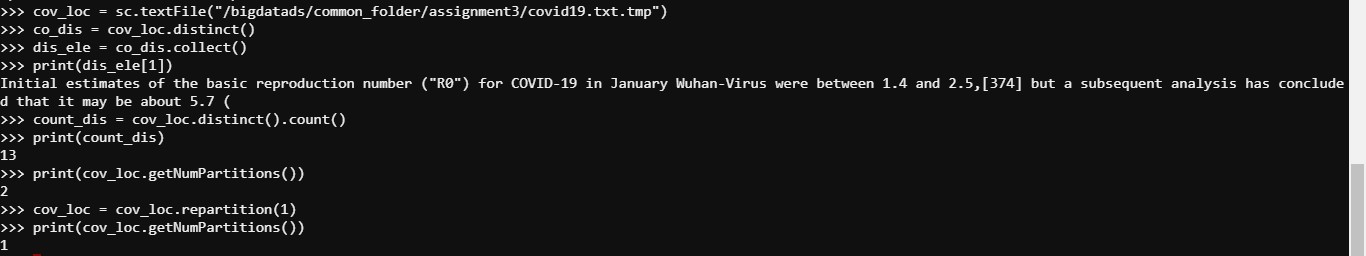
count\_dis=cov\_loc.distinct().count()

print(count\_dis)

print(cov\_loc.getNumPartitions())

cov\_loc=cov\_loc.repartition(1)

print(cov\_loc.getNumPartitions())



Q4 *Dataset: World\_Bank\_Indicators.csv*

The dataset is present at the HDFS path

'hdfs:///bigdatads/common\_folder/assignment3/World\_Bank\_Indicators.csv', contains data dump regarding population, health, internet, GDP, etc. Find the following details:

1. Highest urban population - Country having the highest urban population for the year 2010
2. Most populous countries - List of countries in the descending order of their population

Output Format: Paste the commands used to solve this problem in the text box below. Also, attach a screenshot(s) of the code.

Ans4 fl=sc.textFile("/bigdatads/common\_folder/assignment3/World\_Bank\_Indicators.csv")

import re

fs1=fl.collect()

fs1.pop(0)

for i in range(len(fs1)):

... fs1[i]=re.sub(r'(?!(([^"]\*"){2})\*[^"]\*$),','',fs1[i])

for i in range(len(fs1)):

... fs1[i]=re.sub(r'(["]+)','',fs1[i])

fl1=sc.parallelize(fs1)

filesorted=fl1.map(lambda x:x.split(","))

columns=['Country\_Name','Date1','Transit\_Railways','Transit\_Passenger\_cars','Business\_Mobile\_phone\_subscribers','Business\_Internet\_users','Health\_Mortality\_under\_5','Health\_expenditure\_per\_capita','Health\_expenditure\_total','Population\_Total','Population\_Urban','Population\_Birth\_rate\_crude','Health\_Life\_expectancy\_at\_birth\_female','Health\_Life\_expectancy\_at\_birth\_male','Health\_Life\_expectancy\_at\_birth\_total','Population\_Ages\_0\_14','Population\_Ages\_15\_64','Population\_Ages\_65','Finance\_GDP','Finance\_GDP\_percapita']

wbi=filesorted.toDF(columns)

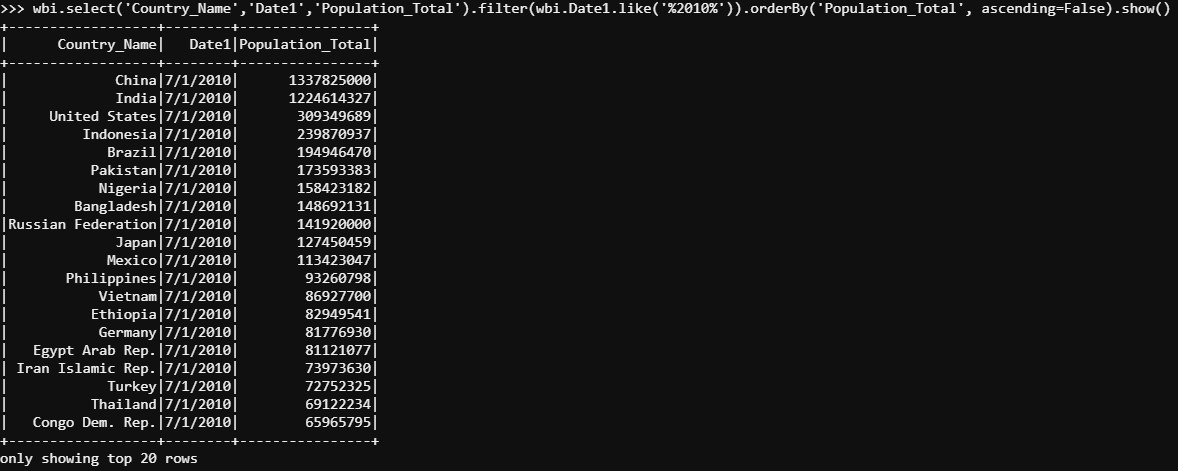
wbi.show()

from pyspark.sql.types import IntegerType, LongType

wbi=wbi.withColumn('Transit\_Railways',wbi['Transit\_Railways'].cast(IntegerType())).withColumn('Transit\_Passenger\_cars',wbi['Transit\_Passenger\_cars'].cast(IntegerType())).withColumn('Business\_Mobile\_phone\_subscribers',wbi['Business\_Mobile\_phone\_subscribers'].cast(IntegerType())).withColumn('Business\_Internet\_users',wbi['Business\_Internet\_users'].cast(IntegerType())).withColumn('Health\_Mortality\_under\_5',wbi['Health\_Mortality\_under\_5'].cast(IntegerType())).withColumn('Health\_expenditure\_per\_capita',wbi['Health\_expenditure\_per\_capita'].cast(IntegerType())).withColumn('Health\_expenditure\_total',wbi['Health\_expenditure\_total'].cast(IntegerType())).withColumn('Population\_Total',wbi['Population\_Total'].cast(IntegerType())).withColumn('Population\_Urban',wbi['Population\_Urban'].cast(IntegerType())).withColumn('Population\_Birth\_rate\_crude',wbi['Population\_Birth\_rate\_crude'].cast(IntegerType())).withColumn('Health\_Life\_expectancy\_at\_birth\_female',wbi['Health\_Life\_expectancy\_at\_birth\_female'].cast(IntegerType())).withColumn('Health\_Life\_expectancy\_at\_birth\_male',wbi['Health\_Life\_expectancy\_at\_birth\_male'].cast(IntegerType())).withColumn('Health\_Life\_expectancy\_at\_birth\_total',wbi['Health\_Life\_expectancy\_at\_birth\_total'].cast(IntegerType())).withColumn('Population\_Ages\_0\_14',wbi['Population\_Ages\_0\_14'].cast(IntegerType())).withColumn('Population\_Ages\_15\_64',wbi['Population\_Ages\_15\_64'].cast(IntegerType())).withColumn('Population\_Ages\_65',wbi['Population\_Ages\_65'].cast(IntegerType())).withColumn('Finance\_GDP',wbi['Finance\_GDP'].cast(LongType())).withColumn('Finance\_GDP\_percapita',wbi['Finance\_GDP\_percapita'].cast(IntegerType()))

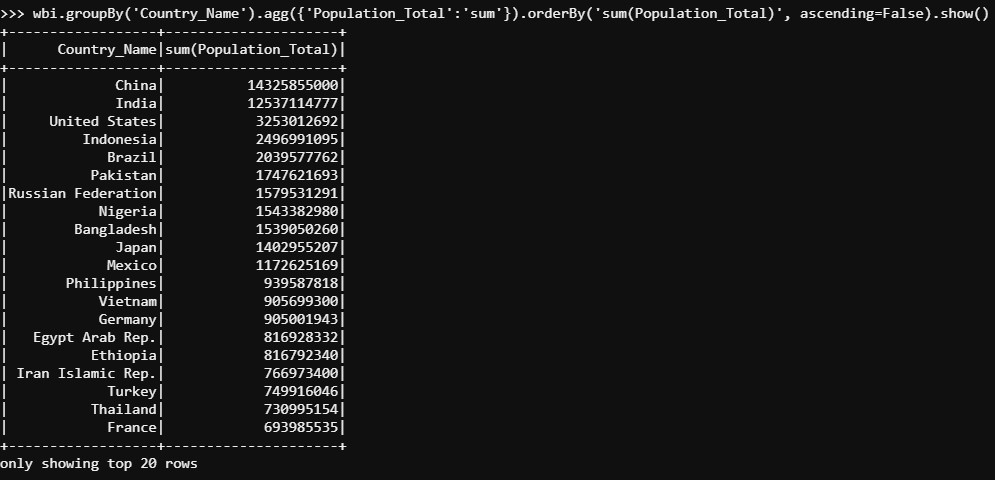
wbi=wbi.na.fill(value=0)

wbi.filter(wbi.Date1.like('%2010%')).agg({'Population\_Urban':'max'}).collect()



ls1=wbi.groupBy('Country\_Name').agg({'Population\_Total':'sum'}).orderBy('sum(Population\_Total)', ascending=False).collect()

print(ls1[0])



Q5 *Dataset: World\_Bank\_Indicators.csv*

With the same given dataset find out the highest population growth, i.e., the country with the highest % population growth in the past decade.

Output Format: Paste the commands used to solve this problem in the text box below. Also, attach a screenshot(s) of the code.

Ans5

ls1=wbi.select('Country\_Name','Date1','Population\_Total').filter(wbi.Date1.like('%2010%')|wbi.Date1.like('%2000%')).collect()

rdd1=sc.parallelize(ls1)

col=['Country\_Name','Date1','Population\_Total']

wb1=rdd1.toDF(col)

wb1.show()

from pyspark.sql import functions as F

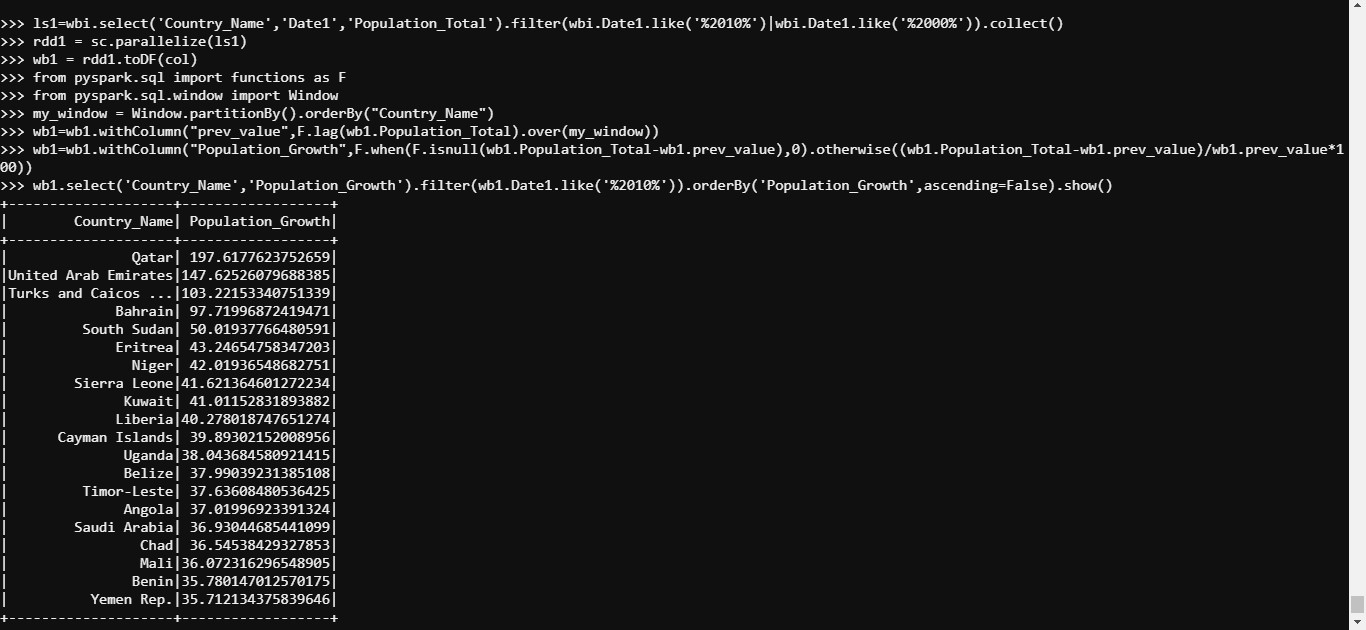
from pyspark.sql.window import Window

my\_window=Window.partitionBy().orderBy("Country\_Name")

wb1=wb1.withColumn("prev\_value",F.lag(wb1.Population\_Total).over(my\_window))

wb1=wb1.withColumn("Population\_Growth",F.when(F.isnull(wb1.Population\_Total-wb1.prev\_value),0).otherwise((wb1.Population\_Total-wb1.prev\_value)/wb1.prev\_value\*100))

wb1.select('Country\_Name','Population\_Growth').filter(wb1.Date1.like('%2010%')).orderBy('Population\_Growth',ascending=False).show()



Q6 *Dataset: World\_Bank\_Indicators.csv*

With the same given dataset find out the highest and lowest GDP growth, i.e., the countries with the highest and lowest GDP growth from 2009 to 2010 in descending order.

Output Format: Paste the commands used to solve this problem in the text box below. Also, attach a screenshot(s) of the code.

Ans6 ls1=wbi.select('Country\_Name','Date1','Finance\_GDP').filter(wbi.Date1.like('%2010%')|wbi.Date1.like('%2009%')).collect()

rdd1=sc.parallelize(ls1)

col=['Country\_Name','Date1','Finance\_GDP']

wb1=rdd1.toDF(col)

wb1.show()

from pyspark.sql import functions as F

from pyspark.sql.window import Window

my\_window=Window.partitionBy().orderBy("Country\_Name")

wb1=wb1.withColumn("prev\_value",F.lag(wb1.Finance\_GDP).over(my\_window))

wb1=wb1.withColumn("GDP\_Growth",F.when(F.isnull(wb1.Finance\_GDP-wb1.prev\_value),0).otherwise(wb1.Finance\_GDP-wb1.prev\_value))

ls2=wb1.select('Country\_Name','GDP\_Growth').filter(wb1.Date1.like('%2010%')).orderBy('GDP\_Growth',ascending=False).collect()

print(ls2[0])

print(ls2[len(ls2)-1])



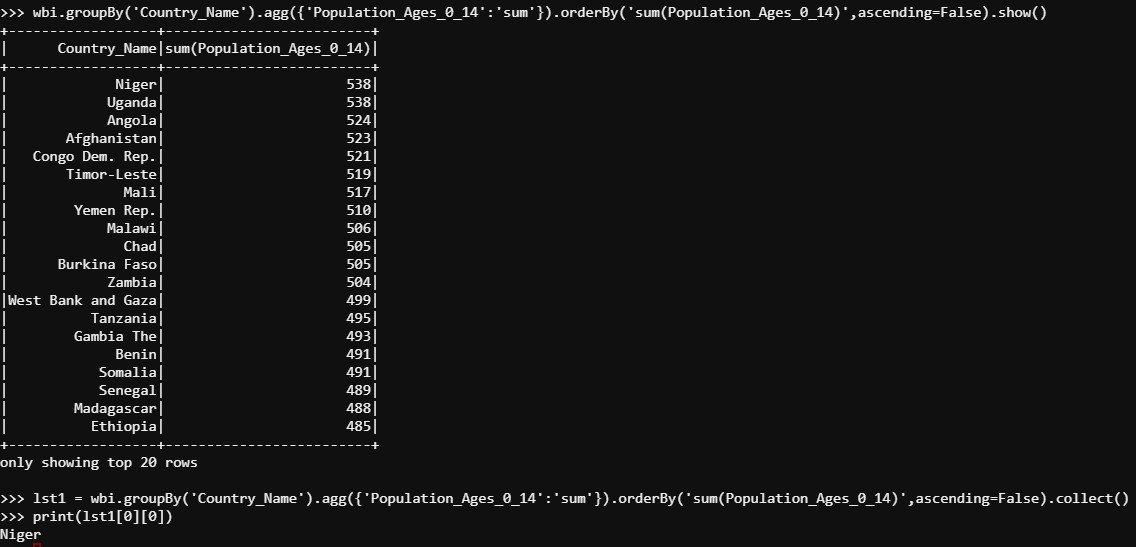
Q7 *Dataset: World\_Bank\_Indicators.csv*

With the same given dataset find out the youngest country, i.e., the yearly distribution of youngest countries.

Output Format: Paste the commands used to solve this problem in the text box below. Also, attach a screenshot(s) of the code.

Ans7 lst1=wbi.groupBy('Country\_Name').agg({'Population\_Ages\_0\_14':'sum'}).orderBy('sum(Population\_Ages\_0\_14)',ascending=False).collect()

print(lst1[0][0])



Q8 *Datasets: car\_data\_test.csv, car\_data\_train.csv*

The dataset is present at the HDFS path

'hdfs:///bigdatads/common\_folder/assignment3/car\_data\_test/train.csv', contains data on used car prices:

* make - machine firm
* model - model of the car
* price USD - price in dollars (target variable)
* year - production year
* condition - represents the condition at the sale moment (with mileage, for parts, etc)
* mileage - mileage in kilometers
* fuel\_type - type of the fuel (electro, petrol, diesel)
* volume of the engine
* color
* transmission
* drive unit

Use the training dataset to train a linear regression model on PySpark by performing the following steps:

1. Read the training dataset

2. Change the datatype of columns: 'priceUSD', 'mileage(kilometers)', 'volume(cm3)' to double

3. Drop 'year', 'model' and 'color' columns

4. Use StringIndexer to map string columns to label indices: 'make', 'condition', 'fuel\_type', 'transmission', 'drive\_unit'

5. Use OneHotEncoder to encode the categorical columns

6. Assemble these features using VectorAssembler

7. Create a base pipeline that does the above 3 steps

8. Create a parameter grid for the following parameters:

* maxIter: [10,15]
* regParam,[0.5, .05]

9. Using CrossValidator fit the training data

10. Evaluate the cvmodel on testing data by using RegressionEvaluator and find the RMSE value up to 5 decimal places

Output Format: Save the code in .ipynb format. Create a folder structure 'assignment3/question8' on your cloud lab and place the .ipynb file in the folder you created.

Mention below, the file name and the path where you have placed the file.

Ans8 <http://bdlabs.edureka.co:50005/user/edureka_1045049/notebooks/assignment3/question8.ipynb>