HDS 5230 High Performance Computing - HW7

March 5, 2019

Author: Miao Cai

0.1 Question 1

In [73]:
```
import os
import numpy as np
import dask.dataframe as dd
import dask.array as da
import dask.delayed
from dask.dot import dot_graph
```

In [74]:
```
op = dd.read_csv('healthcare2/OutpatientVisit.csv')
diseasemap = dd.read_csv('healthcare2/DiseaseMap.csv')

# reshape ICD wide to long
oplong = dd.melt(op, id_vars = 'PatientID',
var_name = 'DiagNum', value_name = 'ICD10',
value_vars = ['ICD10_1', 'ICD10_2', 'ICD10_3'])
```

In [75]:
```
depression = diseasemap.loc[diseasemap.Condition == 'Depression', 'ICD10'].compute()
depression_str = '|'.join(depression)
```

In [9]:
```
# recode multiple depression diagnosis to 1
def f(x):
    if x > 1:
        x = 1
    return x

oplong['depression'] = oplong.ICD10.str.contains(depression_str)
# depression on each patient
q1 = oplong.groupby('PatientID').depression.agg('sum').reset_index()
q1['depression'] = q1.depression.apply(f) # recode
```

```
/Users/miaocai/anaconda3/lib/python3.7/site-packages/dask/dataframe/core.py:2259: UserWarning:
  Before: .apply(func)
  After:  .apply(func, meta={'x': 'f8', 'y': 'f8'}) for dataframe result
  or:     .apply(func, meta=('x', 'f8'))            for series result
  warnings.warn(msg)
```

0.1.1 The answer for Q1 a)

```
In [10]: q1['depression'].sum().compute()/len(q1['depression'])
```

```
Out[10]: 0.10718648208469056
```

0.1.2 The answer for Q1 b)

```
In [11]: def f12(x):
             if x != 0:
                 x = 1
             return x

         Mortality = dd.read_csv("./healthcare2/Mortality.csv")
         p12 = q1.merge(Mortality, how = 'left')
         p12['Death'] = p12.DateOfDeath.fillna(0)
         p12['Death'] = p12['Death'].apply(f12)
         p12.groupby('depression').Death.apply(lambda x: x.sum()/len(x)).reset_index().head()
```

```
/Users/miaocai/anaconda3/lib/python3.7/site-packages/ipykernel_launcher.py:10: UserWarning: `m
  Before: .apply(func)
  After:  .apply(func, meta={'x': 'f8', 'y': 'f8'}) for dataframe result
  or:     .apply(func, meta=('x', 'f8'))            for series result
  # Remove the CWD from sys.path while we load stuff.
/Users/miaocai/anaconda3/lib/python3.7/site-packages/dask/dataframe/core.py:4382: UserWarning:
  warnings.warn(msg.format(n, len(r)))
```

```
Out[11]:    depression     Death
         0           0  0.342834
         1           1  0.430199
```

0.2 Question 2

```
In [79]: from pyspark.sql import SparkSession, DataFrame
         from pyspark.sql.functions import array, col, explode, \
         lit, struct,concat,regexp_extract
         from typing import Iterable
         import numpy as np
         import pandas as pd

         spark = SparkSession.builder\
                 .appName("Data Import")\
                 .master("local[1]")\
                 .getOrCreate()
```

```
In [94]: op = spark.read.csv(
             'healthcare2/OutpatientVisit.csv',
             header=True, inferSchema=True)
```

```
Disease = spark.read.csv(
    'healthcare2/DiseaseMap.csv',
    header=True, inferSchema=True)

Mortality = spark.read.csv(
    'healthcare2/Mortality.csv',
    header=True, inferSchema=True)
```

```
In [95]: def melt(
             df: DataFrame,
             id_vars: Iterable[str], value_vars: Iterable[str],
             var_name: str="variable", value_name: str="value") -> DataFrame:
         """Convert :class:`DataFrame` from wide to long format."""

         # Create array<struct<variable: str, value: ...>>
         _vars_and_vals = array(*(
             struct(lit(c).alias(var_name), col(c).alias(value_name))
             for c in value_vars))

         # Add to the DataFrame and explode
         _tmp = df.withColumn("_vars_and_vals", explode(_vars_and_vals))

         cols = id_vars + [
                 col("_vars_and_vals")[x].alias(x) for x in [var_name, value_name]]
         return _tmp.select(*cols)


df2=melt(op, id_vars=['PatientID'], value_vars=['ICD10_1', 'ICD10_2','ICD10_3'])
```

```
In [113]: df2 = df2.withColumnRenamed('value','ICD10')
          df3 = df2.join(Disease, on='ICD10',
                         how='left_outer')
```

```
In [109]: df4 = df3.filter(df3['Condition'] == 'Depression').select('PatientID').distinct()
```

```
---------------------------------------------------------------------------

Py4JJavaError                             Traceback (most recent call last)

<ipython-input-109-7996833657c7> in <module>
----> 1 df4.show(5)


~/anaconda3/lib/python3.7/site-packages/pyspark/sql/dataframe.py in show(self, n, trun
376         """
377         if isinstance(truncate, bool) and truncate:
--> 378             print(self._jdf.showString(n, 20, vertical))
```

```
    379         else:
    380             print(self._jdf.showString(n, int(truncate), vertical))


    ~/anaconda3/lib/python3.7/site-packages/py4j/java_gateway.py in __call__(self, *args)
   1255         answer = self.gateway_client.send_command(command)
   1256         return_value = get_return_value(
-> 1257             answer, self.gateway_client, self.target_id, self.name)
   1258
   1259         for temp_arg in temp_args:


    ~/anaconda3/lib/python3.7/site-packages/pyspark/sql/utils.py in deco(*a, **kw)
     61     def deco(*a, **kw):
     62         try:
---> 63             return f(*a, **kw)
     64         except py4j.protocol.Py4JJavaError as e:
     65             s = e.java_exception.toString()


    ~/anaconda3/lib/python3.7/site-packages/py4j/protocol.py in get_return_value(answer, ga
    326                 raise Py4JJavaError(
    327                     "An error occurred while calling {0}{1}{2}.\n".
--> 328                     format(target_id, ".", name), value)
    329             else:
    330                 raise Py4JError(


    Py4JJavaError: An error occurred while calling o1810.showString.
: org.apache.spark.SparkException: Exception thrown in awaitResult:
        at org.apache.spark.util.ThreadUtils$.awaitResult(ThreadUtils.scala:226)
        at org.apache.spark.sql.execution.exchange.BroadcastExchangeExec.doExecuteBroadcast
        at org.apache.spark.sql.execution.InputAdapter.doExecuteBroadcast(WholeStageCodeger
        at org.apache.spark.sql.execution.SparkPlan$$anonfun$executeBroadcast$1.apply(Spark
        at org.apache.spark.sql.execution.SparkPlan$$anonfun$executeBroadcast$1.apply(Spark
        at org.apache.spark.sql.execution.SparkPlan$$anonfun$executeQuery$1.apply(SparkPlan
        at org.apache.spark.rdd.RDDOperationScope$.withScope(RDDOperationScope.scala:151)
        at org.apache.spark.sql.execution.SparkPlan.executeQuery(SparkPlan.scala:152)
        at org.apache.spark.sql.execution.SparkPlan.executeBroadcast(SparkPlan.scala:140)
        at org.apache.spark.sql.execution.joins.BroadcastHashJoinExec.prepareBroadcast(Broa
        at org.apache.spark.sql.execution.joins.BroadcastHashJoinExec.codegenInner(Broadcas
        at org.apache.spark.sql.execution.joins.BroadcastHashJoinExec.doConsume(BroadcastHa
        at org.apache.spark.sql.execution.CodegenSupport$class.consume(WholeStageCodegenExe
        at org.apache.spark.sql.execution.ProjectExec.consume(basicPhysicalOperators.scala
        at org.apache.spark.sql.execution.ProjectExec.doConsume(basicPhysicalOperators.scal
        at org.apache.spark.sql.execution.CodegenSupport$class.consume(WholeStageCodegenExe
        at org.apache.spark.sql.execution.FilterExec.consume(basicPhysicalOperators.scala:8
        at org.apache.spark.sql.execution.FilterExec.doConsume(basicPhysicalOperators.scala
```

```
at org.apache.spark.sql.execution.CodegenSupport$class.consume(WholeStageCodegenExe
at org.apache.spark.sql.execution.InputAdapter.consume(WholeStageCodegenExec.scala
at org.apache.spark.sql.execution.InputAdapter.doProduce(WholeStageCodegenExec.scal
at org.apache.spark.sql.execution.CodegenSupport$$anonfun$produce$1.apply(WholeStag
at org.apache.spark.sql.execution.CodegenSupport$$anonfun$produce$1.apply(WholeStag
at org.apache.spark.sql.execution.SparkPlan$$anonfun$executeQuery$1.apply(SparkPlan
at org.apache.spark.rdd.RDDOperationScope$.withScope(RDDOperationScope.scala:151)
at org.apache.spark.sql.execution.SparkPlan.executeQuery(SparkPlan.scala:152)
at org.apache.spark.sql.execution.CodegenSupport$class.produce(WholeStageCodegenExe
at org.apache.spark.sql.execution.InputAdapter.produce(WholeStageCodegenExec.scala
at org.apache.spark.sql.execution.FilterExec.doProduce(basicPhysicalOperators.scala
at org.apache.spark.sql.execution.CodegenSupport$$anonfun$produce$1.apply(WholeStag
at org.apache.spark.sql.execution.CodegenSupport$$anonfun$produce$1.apply(WholeStag
at org.apache.spark.sql.execution.SparkPlan$$anonfun$executeQuery$1.apply(SparkPlan
at org.apache.spark.rdd.RDDOperationScope$.withScope(RDDOperationScope.scala:151)
at org.apache.spark.sql.execution.SparkPlan.executeQuery(SparkPlan.scala:152)
at org.apache.spark.sql.execution.CodegenSupport$class.produce(WholeStageCodegenExe
at org.apache.spark.sql.execution.FilterExec.produce(basicPhysicalOperators.scala:8
at org.apache.spark.sql.execution.ProjectExec.doProduce(basicPhysicalOperators.scal
at org.apache.spark.sql.execution.CodegenSupport$$anonfun$produce$1.apply(WholeStag
at org.apache.spark.sql.execution.CodegenSupport$$anonfun$produce$1.apply(WholeStag
at org.apache.spark.sql.execution.SparkPlan$$anonfun$executeQuery$1.apply(SparkPlan
at org.apache.spark.rdd.RDDOperationScope$.withScope(RDDOperationScope.scala:151)
at org.apache.spark.sql.execution.SparkPlan.executeQuery(SparkPlan.scala:152)
at org.apache.spark.sql.execution.CodegenSupport$class.produce(WholeStageCodegenExe
at org.apache.spark.sql.execution.ProjectExec.produce(basicPhysicalOperators.scala
at org.apache.spark.sql.execution.joins.BroadcastHashJoinExec.doProduce(BroadcastHa
at org.apache.spark.sql.execution.CodegenSupport$$anonfun$produce$1.apply(WholeStag
at org.apache.spark.sql.execution.CodegenSupport$$anonfun$produce$1.apply(WholeStag
at org.apache.spark.sql.execution.SparkPlan$$anonfun$executeQuery$1.apply(SparkPlan
at org.apache.spark.rdd.RDDOperationScope$.withScope(RDDOperationScope.scala:151)
at org.apache.spark.sql.execution.SparkPlan.executeQuery(SparkPlan.scala:152)
at org.apache.spark.sql.execution.CodegenSupport$class.produce(WholeStageCodegenExe
at org.apache.spark.sql.execution.joins.BroadcastHashJoinExec.produce(BroadcastHash
at org.apache.spark.sql.execution.ProjectExec.doProduce(basicPhysicalOperators.scal
at org.apache.spark.sql.execution.CodegenSupport$$anonfun$produce$1.apply(WholeStag
at org.apache.spark.sql.execution.CodegenSupport$$anonfun$produce$1.apply(WholeStag
at org.apache.spark.sql.execution.SparkPlan$$anonfun$executeQuery$1.apply(SparkPlan
at org.apache.spark.rdd.RDDOperationScope$.withScope(RDDOperationScope.scala:151)
at org.apache.spark.sql.execution.SparkPlan.executeQuery(SparkPlan.scala:152)
at org.apache.spark.sql.execution.CodegenSupport$class.produce(WholeStageCodegenExe
at org.apache.spark.sql.execution.ProjectExec.produce(basicPhysicalOperators.scala
at org.apache.spark.sql.execution.aggregate.HashAggregateExec.doProduceWithKeys(Has
at org.apache.spark.sql.execution.aggregate.HashAggregateExec.doProduce(HashAggrega
at org.apache.spark.sql.execution.CodegenSupport$$anonfun$produce$1.apply(WholeStag
at org.apache.spark.sql.execution.CodegenSupport$$anonfun$produce$1.apply(WholeStag
at org.apache.spark.sql.execution.SparkPlan$$anonfun$executeQuery$1.apply(SparkPlan
at org.apache.spark.rdd.RDDOperationScope$.withScope(RDDOperationScope.scala:151)
```

```
at org.apache.spark.sql.execution.SparkPlan.executeQuery(SparkPlan.scala:152)
at org.apache.spark.sql.execution.CodegenSupport$class.produce(WholeStageCodegenExe
at org.apache.spark.sql.execution.aggregate.HashAggregateExec.produce(HashAggregate
at org.apache.spark.sql.execution.WholeStageCodegenExec.doCodeGen(WholeStageCodegen
at org.apache.spark.sql.execution.WholeStageCodegenExec.doExecute(WholeStageCodegen
at org.apache.spark.sql.execution.SparkPlan$$anonfun$execute$1.apply(SparkPlan.scal
at org.apache.spark.sql.execution.SparkPlan$$anonfun$execute$1.apply(SparkPlan.scal
at org.apache.spark.sql.execution.SparkPlan$$anonfun$executeQuery$1.apply(SparkPlan
at org.apache.spark.rdd.RDDOperationScope$.withScope(RDDOperationScope.scala:151)
at org.apache.spark.sql.execution.SparkPlan.executeQuery(SparkPlan.scala:152)
at org.apache.spark.sql.execution.SparkPlan.execute(SparkPlan.scala:127)
at org.apache.spark.sql.execution.exchange.ShuffleExchangeExec.prepareShuffleDepend
at org.apache.spark.sql.execution.exchange.ShuffleExchangeExec$$anonfun$doExecute$1
at org.apache.spark.sql.execution.exchange.ShuffleExchangeExec$$anonfun$doExecute$1
at org.apache.spark.sql.catalyst.errors.package$.attachTree(package.scala:52)
at org.apache.spark.sql.execution.exchange.ShuffleExchangeExec.doExecute(ShuffleExe
at org.apache.spark.sql.execution.SparkPlan$$anonfun$execute$1.apply(SparkPlan.scal
at org.apache.spark.sql.execution.SparkPlan$$anonfun$execute$1.apply(SparkPlan.scal
at org.apache.spark.sql.execution.SparkPlan$$anonfun$executeQuery$1.apply(SparkPlan
at org.apache.spark.rdd.RDDOperationScope$.withScope(RDDOperationScope.scala:151)
at org.apache.spark.sql.execution.SparkPlan.executeQuery(SparkPlan.scala:152)
at org.apache.spark.sql.execution.SparkPlan.execute(SparkPlan.scala:127)
at org.apache.spark.sql.execution.InputAdapter.inputRDDs(WholeStageCodegenExec.scal
at org.apache.spark.sql.execution.aggregate.HashAggregateExec.inputRDDs(HashAggrega
at org.apache.spark.sql.execution.WholeStageCodegenExec.doExecute(WholeStageCodegen
at org.apache.spark.sql.execution.SparkPlan$$anonfun$execute$1.apply(SparkPlan.scal
at org.apache.spark.sql.execution.SparkPlan$$anonfun$execute$1.apply(SparkPlan.scal
at org.apache.spark.sql.execution.SparkPlan$$anonfun$executeQuery$1.apply(SparkPlan
at org.apache.spark.rdd.RDDOperationScope$.withScope(RDDOperationScope.scala:151)
at org.apache.spark.sql.execution.SparkPlan.executeQuery(SparkPlan.scala:152)
at org.apache.spark.sql.execution.SparkPlan.execute(SparkPlan.scala:127)
at org.apache.spark.sql.execution.SparkPlan.getByteArrayRdd(SparkPlan.scala:247)
at org.apache.spark.sql.execution.SparkPlan.executeTake(SparkPlan.scala:339)
at org.apache.spark.sql.execution.CollectLimitExec.executeCollect(limit.scala:38)
at org.apache.spark.sql.Dataset.org$apache$spark$sql$Dataset$$collectFromPlan(Datas
at org.apache.spark.sql.Dataset$$anonfun$head$1.apply(Dataset.scala:2545)
at org.apache.spark.sql.Dataset$$anonfun$head$1.apply(Dataset.scala:2545)
at org.apache.spark.sql.Dataset$$anonfun$53.apply(Dataset.scala:3365)
at org.apache.spark.sql.execution.SQLExecution$$anonfun$withNewExecutionId$1.apply
at org.apache.spark.sql.execution.SQLExecution$.withSQLConfPropagated(SQLExecution
at org.apache.spark.sql.execution.SQLExecution$.withNewExecutionId(SQLExecution.sca
at org.apache.spark.sql.Dataset.withAction(Dataset.scala:3364)
at org.apache.spark.sql.Dataset.head(Dataset.scala:2545)
at org.apache.spark.sql.Dataset.take(Dataset.scala:2759)
at org.apache.spark.sql.Dataset.getRows(Dataset.scala:255)
at org.apache.spark.sql.Dataset.showString(Dataset.scala:292)
at jdk.internal.reflect.GeneratedMethodAccessor64.invoke(Unknown Source)
at java.base/jdk.internal.reflect.DelegatingMethodAccessorImpl.invoke(DelegatingMet
```

```
        at java.base/java.lang.reflect.Method.invoke(Method.java:566)
        at py4j.reflection.MethodInvoker.invoke(MethodInvoker.java:244)
        at py4j.reflection.ReflectionEngine.invoke(ReflectionEngine.java:357)
        at py4j.Gateway.invoke(Gateway.java:282)
        at py4j.commands.AbstractCommand.invokeMethod(AbstractCommand.java:132)
        at py4j.commands.CallCommand.execute(CallCommand.java:79)
        at py4j.GatewayConnection.run(GatewayConnection.java:238)
        at java.base/java.lang.Thread.run(Thread.java:834)
Caused by: java.lang.IllegalArgumentException: Unsupported class file major version 55
        at org.apache.xbean.asm6.ClassReader.<init>(ClassReader.java:166)
        at org.apache.xbean.asm6.ClassReader.<init>(ClassReader.java:148)
        at org.apache.xbean.asm6.ClassReader.<init>(ClassReader.java:136)
        at org.apache.xbean.asm6.ClassReader.<init>(ClassReader.java:237)
        at org.apache.spark.util.ClosureCleaner$.getClassReader(ClosureCleaner.scala:49)
        at org.apache.spark.util.FieldAccessFinder$$anon$3$$anonfun$visitMethodInsn$2.apply
        at org.apache.spark.util.FieldAccessFinder$$anon$3$$anonfun$visitMethodInsn$2.apply
        at scala.collection.TraversableLike$WithFilter$$anonfun$foreach$1.apply(Traversable
        at scala.collection.mutable.HashMap$$anon$1$$anonfun$foreach$2.apply(HashMap.scala
        at scala.collection.mutable.HashMap$$anon$1$$anonfun$foreach$2.apply(HashMap.scala
        at scala.collection.mutable.HashTable$class.foreachEntry(HashTable.scala:236)
        at scala.collection.mutable.HashMap.foreachEntry(HashMap.scala:40)
        at scala.collection.mutable.HashMap$$anon$1.foreach(HashMap.scala:134)
        at scala.collection.TraversableLike$WithFilter.foreach(TraversableLike.scala:732)
        at org.apache.spark.util.FieldAccessFinder$$anon$3.visitMethodInsn(ClosureCleaner.s
        at org.apache.xbean.asm6.ClassReader.readCode(ClassReader.java:2175)
        at org.apache.xbean.asm6.ClassReader.readMethod(ClassReader.java:1238)
        at org.apache.xbean.asm6.ClassReader.accept(ClassReader.java:631)
        at org.apache.xbean.asm6.ClassReader.accept(ClassReader.java:355)
        at org.apache.spark.util.ClosureCleaner$$anonfun$org$apache$spark$util$ClosureClean
        at org.apache.spark.util.ClosureCleaner$$anonfun$org$apache$spark$util$ClosureClean
        at scala.collection.immutable.List.foreach(List.scala:392)
        at org.apache.spark.util.ClosureCleaner$.org$apache$spark$util$ClosureCleaner$$clea
        at org.apache.spark.util.ClosureCleaner$.clean(ClosureCleaner.scala:162)
        at org.apache.spark.SparkContext.clean(SparkContext.scala:2326)
        at org.apache.spark.SparkContext.runJob(SparkContext.scala:2100)
        at org.apache.spark.SparkContext.runJob(SparkContext.scala:2126)
        at org.apache.spark.rdd.RDD$$anonfun$collect$1.apply(RDD.scala:945)
        at org.apache.spark.rdd.RDDOperationScope$.withScope(RDDOperationScope.scala:151)
        at org.apache.spark.rdd.RDDOperationScope$.withScope(RDDOperationScope.scala:112)
        at org.apache.spark.rdd.RDD.withScope(RDD.scala:363)
        at org.apache.spark.rdd.RDD.collect(RDD.scala:944)
        at org.apache.spark.sql.execution.SparkPlan.executeCollectIterator(SparkPlan.scala
        at org.apache.spark.sql.execution.exchange.BroadcastExchangeExec$$anonfun$relation
        at org.apache.spark.sql.execution.exchange.BroadcastExchangeExec$$anonfun$relation
        at org.apache.spark.sql.execution.SQLExecution$$anonfun$withExecutionId$1.apply(SQ
        at org.apache.spark.sql.execution.SQLExecution$.withSQLConfPropagated(SQLExecution
        at org.apache.spark.sql.execution.SQLExecution$.withExecutionId(SQLExecution.scala
        at org.apache.spark.sql.execution.exchange.BroadcastExchangeExec$$anonfun$relation
```

```
        at org.apache.spark.sql.execution.exchange.BroadcastExchangeExec$$anonfun$relationF
        at scala.concurrent.impl.Future$PromiseCompletingRunnable.liftedTree1$1(Future.scal
        at scala.concurrent.impl.Future$PromiseCompletingRunnable.run(Future.scala:24)
        at java.base/java.util.concurrent.ThreadPoolExecutor.runWorker(ThreadPoolExecutor.j
        at java.base/java.util.concurrent.ThreadPoolExecutor$Worker.run(ThreadPoolExecutor
        ... 1 more
```

```
In [ ]: df4.select('PatientID').distinct().count()
        #0.10347109120521172
```

```
In [ ]: df2 = df2.withColumnRenamed('value','ICD10')

        df3 = df2.join(Disease,
                       on='ICD10',
                       how='left_outer')

        df3.show(5)
        df4=df3\
            .filter(df3['Condition'] == 'Depression')
        df4.show(5)
        nume=df4.select('PatientID').distinct().count()
        #0.10347109120521172


        #risk of depression among people
        nume/deno
```

```
In [ ]: ## 2 (b)
        df5 = df3.join(Mortality,
                       on='PatientID',
                       how='left_outer')
        df5.show(5)
        df5 = df5.withColumn("mortality", lit(1))
        df5=df5.withColumn("mortality", s_f.when(df5.DateOfDeath.isNull(), lit(0)).otherwise(li
        df6=df5\
            .filter(df5['Condition'] == 'Depression')
        df6.show(5)
        DD=df6.filter(df6['mortality'] != 0).select('PatientID').distinct().count()
        DN=df6.filter(df6['mortality'] == 0).select('PatientID').distinct().count()

        df7=df5\
            .filter(df5['Condition'] != 'Depression')
        ND=df7.filter(df7['mortality'] != 0).select('PatientID').distinct().count()
        NN=df7.filter(df7['mortality'] == 0).select('PatientID').distinct().count()

        #risk of death among depression
```

```
DD/(DD+DN)
 # 0.10347109120521172
#risk of death among non-depression
ND/(ND+NN)
#0.10347109120521172
```