作业7 mapreduce之数据挖掘算法-knn算法

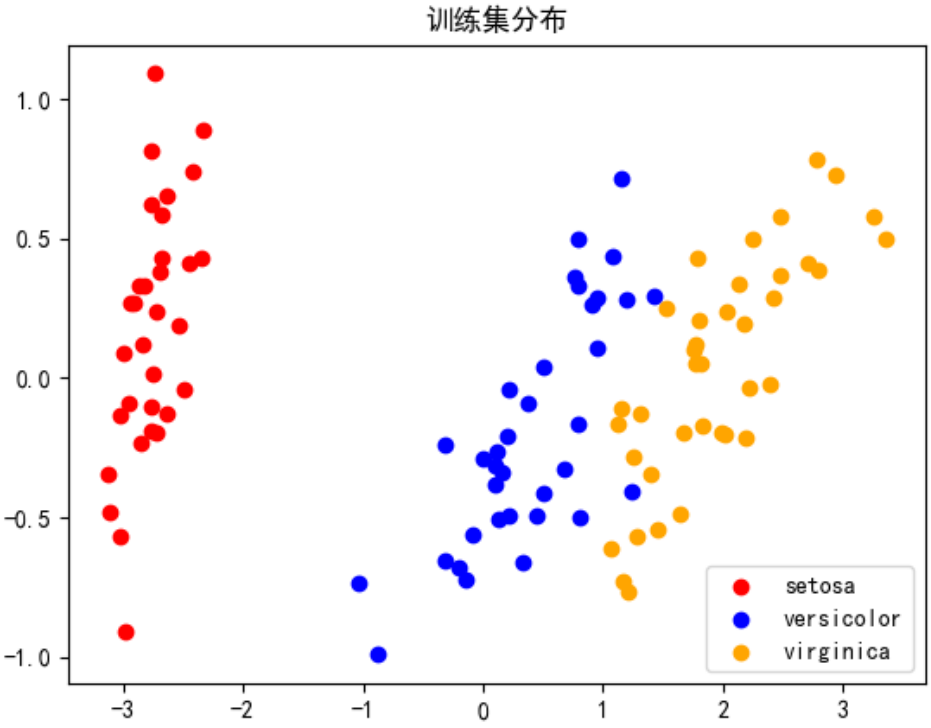
一、解决数据挖掘问题的总体思路

CRISP-DM跨行业数据挖掘标准流程包括：业务理解、数据理解、数据准备、建立模型、模型评价、模型实施。将此次作业代入即

1. 根据已有的鸢尾花量化数据对未知类型的鸢尾花进行类型预测
2. 观察鸢尾花数据，可知此次数据共有4个double型的属性和1个字符串类型的label
3. 根据数据特征处理对其进行预处理，观察此次数据，
    - 类型分布均匀，不存在数据不平衡问题
    - 属性数据完成，无缺失值
    - 由readme并不能断定其由噪音数据，所以忽略对噪音数据的处理

    因此不需要对数据本身进行预处理，只需要在读取文本文件中的数据时，注意属性和label值的数据类型。

    另外，通过将原始数据集使用python进行PCA降到二维数据之后，将数据分布使用scatter函数绘制出如下所示，可以看到类别1和类别2，3的距离较远，类别2，3的距离较近，因此可以推断，该算法对类别1的预测应当更为准确，类别2、3的预测可能会出现个别错误。



4. 由第三步的观察可知，属性值是double类型值，且是连续的，所以在此我选择使用KNN算法，因为连续的属性double值，计算数据之间的距离非常方便，所以此次建模选择KNN算法进行建模。
5. 对于模型效果的评价，根据题目要求采用accuracy
6. 模型实施思路为，在map节点找出离预测值最近的K个点的label将其传入reduce节点，由reduce节点判断哪个label值出现的次数最多，将其设为预测的label。

二、数据挖掘代码体系

数据挖掘体系为以下五个部分，分别对应此代码中的部分为：

原始数据库：input中的train.txt 和predict.txt文件

挖掘前处理模块：读出数据位于KNNMap类和KNNReduce类中的set up函数中，仅执行一次

挖掘操作模块：位于KNNMap类和KNNReduce类的map，reduce函数中

模式评估模块：位于reduce函数中

知识输出模块：预测结果位于reduce函数中，可视化处理见第四部分。

三、代码编写部分

基于书本代码编写，根据题目要求主要修改的地方有

1、改变添加缓存文件的方式，将原始数据添加到trainSet和predictSet中，供map和reduce函数使用。

`kNNJob.addCacheFile(new Path(remainingArgs[2]).toUri());` 70%的训练数据，用于计算，在map中使用

`kNNJob.addCacheFile(new Path(remainingArgs[0]).toUri());` 30%的预测数据（计算accuracy），在reduce中用于计算accuracy

map中set up函数文件 读取训练数据

```
        conf = context.getConfiguration();
    •   trainSet = new *ArrayList*<*Instance*>();
    •   *URI*[] patternsURIs = Job.getInstance(conf).getCacheFiles();
    •   *Path* patternsPath1 = new Path(patternsURIs[0].getPath());
    •   *String* patternsFileName1 = patternsPath1.getName().toString();
    •   *BufferedReader* br = null;
    •   *String* line;
    •   br = new BufferedReader(new FileReader(patternsFileName1));
    •   while((line = br.readLine()) != null){
    •       *Instance* trainInstance = new Instance(line);
    •        trainSet.add(trainInstance);
        }
```

reduce中set up函数与map中类似，将

`Path patternsPath1 = new Path(patternsURIs[0].getPath());` 改为 `Path patternsPath1 = new Path(patternsURIs[1].getPath());`

2、根据鸢尾花的数据格式修改instance中的分词功能、

- instance类用于将原文件中每行数据划分为属性值和label标签，内有getAtrributeValue()，getLable()函数用于取值，由于鸢尾花数据是csv格式，行数据中用“，”间隔，所以将instance函数中分词改为 `String[] value = line.split(",");`
- 由于label是字符串数据，所以label函数改为

```
private String lable;
public String getLable(){
return lable;
}
```

3、根据鸢尾花数据label的类型修改map和reduce函数

KNNMap类：

- 由于代码的实例中，label是double型，而鸢尾花数据的label是字符串，因此需要继承的类为 `Mapper<LongWritable,Text,LongWritable,ListWritable<Text>>`
- 在map函数中，将距离最近的k个点的label写入键值对的value中时，首先需要一个数据类型为Text的arraylist，即 `ArrayList<Text> trainLable = new ArrayList<Text>(k);` 用于存储需要写入的k个label字符串。
- 另外，为了将该list写入键值对传给reduce节点处理，还需要一个listWritable类型，元素为Text的label，即 `ListWritable<Text> lables = new ListWritable<Text>(Text.class);`
- 对应的添加函数改为 `trainLable.add(new Text(trainSet.get(i).getLable()));`

KNNReduce类：

- 由于map节点传入的键值对中的value为ListWritable(Text.class)类型，所以继承的类为 `Reducer<LongWritable,ListWritable<Text>,NullWritable,Text>`
- 同时，reduce函数的迭代类型为 `Iterable<ListWritable<Text>>`
- 由于最终的结果为字符串，因此reduce函数的value类型为Text，所以创建 `Text predictedLable = new Text();`
- reduce类需要计算value中每个label出现的频率，所以需要改 valueOfMostFrequent(val)函数，
    - 函数定义改为 `public Text valueOfMostFrequent(ListWritable<Text> list)`

- 传入的value类型为Text，因此该函数中的Hashmap改为 `HashMap<Text,Integer> tmp = new HashMap<Text,Integer>();`
- 求最大频率对应label的迭代器类型为 `Iterator<Entry<Text, Integer>> iter = tmp.entrySet().iterator();`

4、添加计算accuracy的功能并且将accuracy输入到output文件中

- 总体思路为

  KNNReduce类中：

  set up：input/predict.txt --> PredictSet （ *List* < *String* > PredictSet = new *ArrayList* < *String* >();)

  reduce：已预测总数量sum = 已预测总数量sum + 1

  if (预测正确) right = right + 1

  if (sum == 45) {

  double accuracy = Double.valueOf(right) / Double.valueOf(sum);

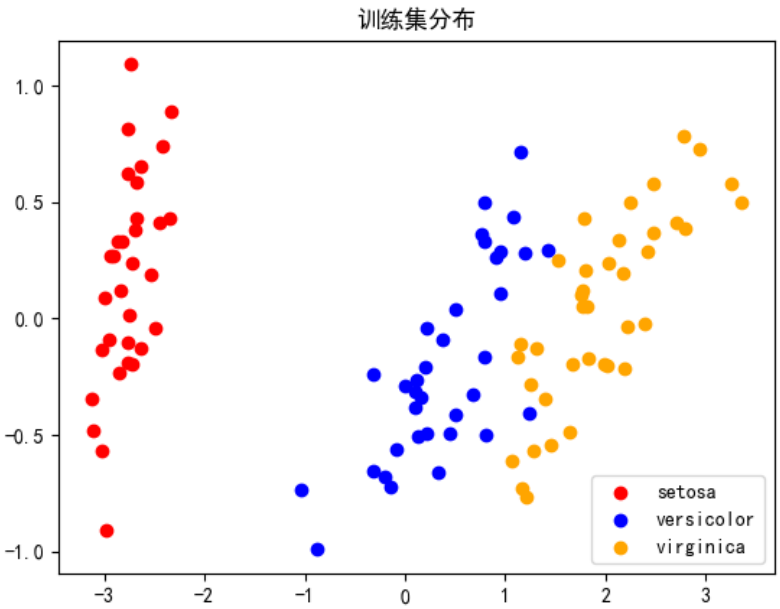
  context.write(NullWritable.get(),new Text(String.valueOf(accuracy)));}

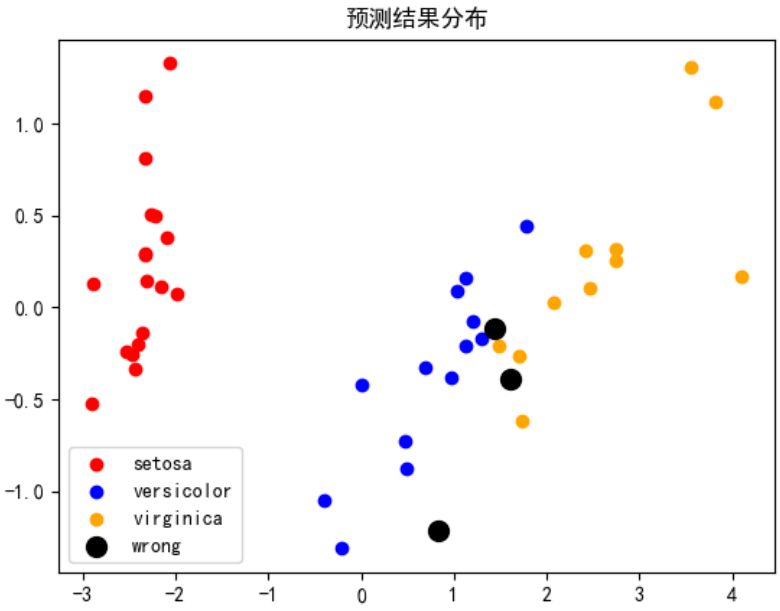
四、数据挖掘结果可视化处理

由于该数据集的特征有4个，所以我采用降维然后画图的方式进行可视化，首先使用PCA将数据降成二维数据，再使用scatter函数画二维分布图。

共有两张图，第一张图用于分析训练集分布，用于大致明白数据的分布。便于选择分类算法。

第二张图是对预测结果进行可视化，将分类错误的点标为黑色，且放大，效果如图所示。（这里以K=3为例）

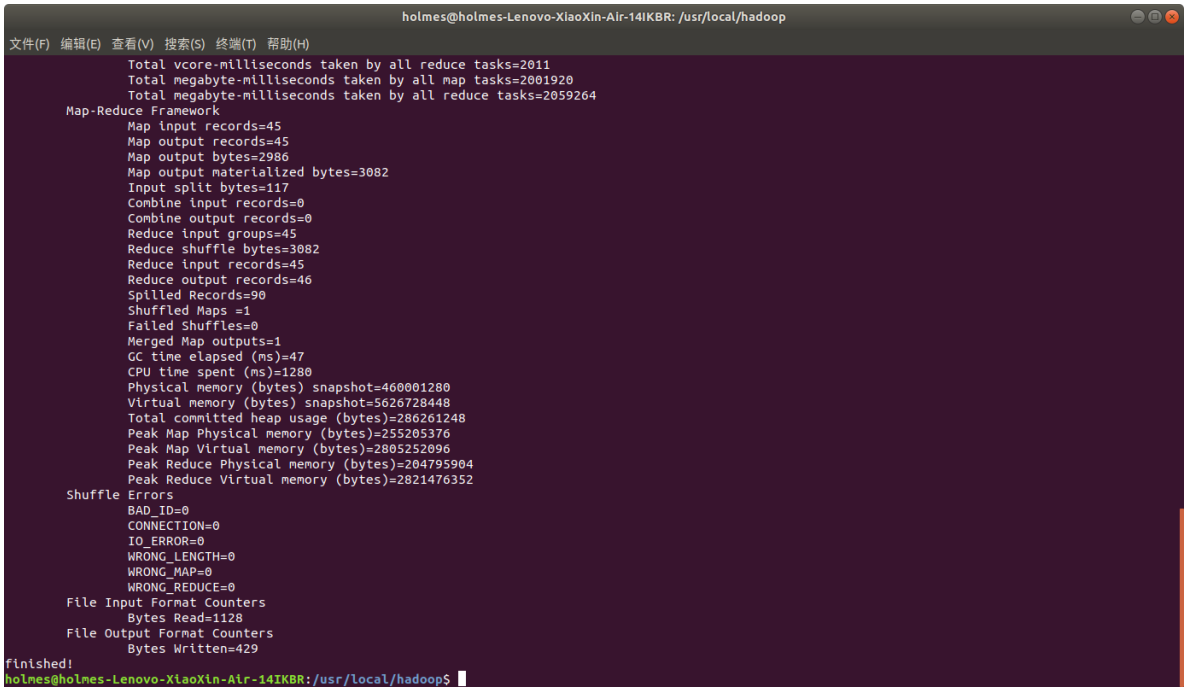


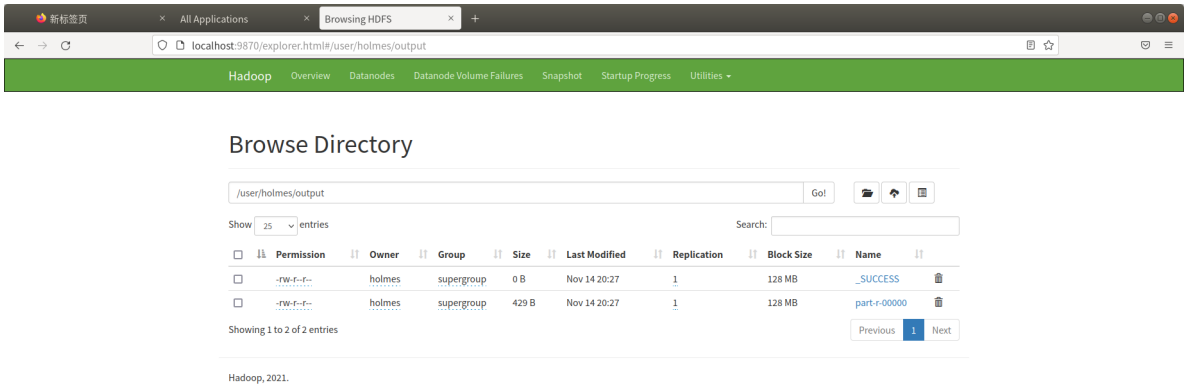


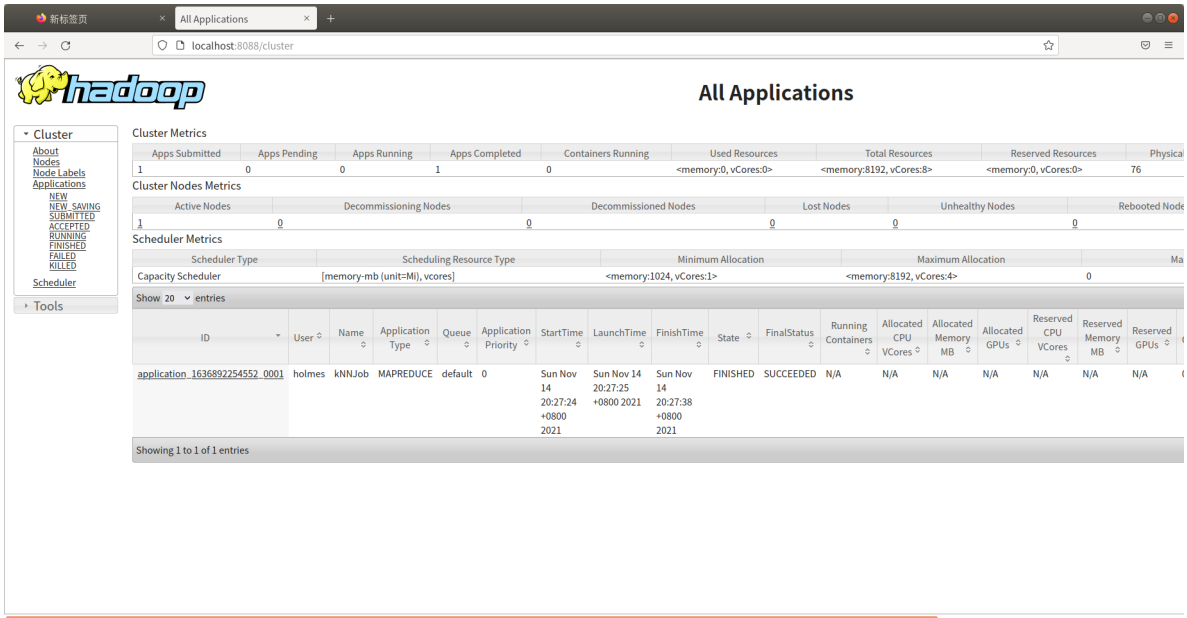
五、结果分析与改进

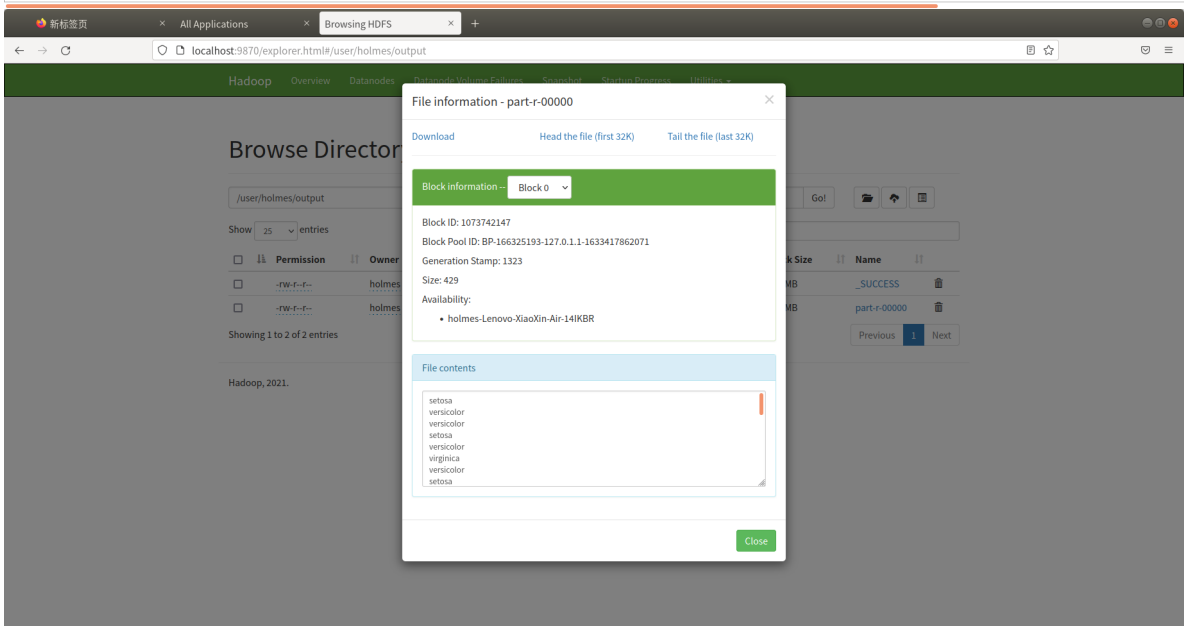
1. 结果分析
    - K = 2，3，4时，准确率均为 42/45 , k = 10时，准确率为43/45
    - 预测错误的数据都为类别2或者类型3，因为距离过近，所以使用KNN算法并不能保证百分之百的分类正确、
2. 改进思路
    - 数据预处理：这里是通过肉眼观察数据来判断数据是否需要进行预处理，没有实现数据处理的自动化，比如使用代码判断是否存在数据不平衡问题，如果存在，则可以在使用KNN进行分类时，给每个类别不同的权重用于提高预测的准确性。比如判断是否存在噪声数据，可以在进行训练之前去除噪声数据。
    - 可以提高扩展性，比如让用户根据自己的数据和目标设置不同类型的准确率。
    - 可以结合不同的分类函数进行分类，比如说，knn算法对分离类别1更为精确，那么可以先使用KNN分离类别1，再使用贝叶斯分布或者决策树进行类别2，3的分类。

六、运行截图


holmes@holmes-Lenovo-XiaoXin-Air-14IKBR: /usr/local/hadoop
文件(F) 编辑(E) 查看(V) 搜索(S) 终端(T) 帮助(H)
Total vcore-milliseconds taken by all reduce tasks=2011
Total megabyte-milliseconds taken by all map tasks=2001920
Total megabyte-milliseconds taken by all reduce tasks=2059264
Map-Reduce Framework
Map input records=45
Map output records=45
Map output bytes=2986
Map output materialized bytes=3082
Input split bytes=117
Combine input records=0
Combine output records=0
Reduce input groups=45
Reduce shuffle bytes=3082
Reduce input records=45
Reduce output records=46
Spilled Records=90
Shuffled Maps =1
Failed Shuffles=0
Merged Map outputs=1
GC time elapsed (ms)=47
CPU time spent (ms)=1280
Physical memory (bytes) snapshot=460001280
Virtual memory (bytes) snapshot=5626728448
Total committed heap usage (bytes)=286261248
Peak Map Physical memory (bytes)=255205376
Peak Map Virtual memory (bytes)=2805252096
Peak Reduce Physical memory (bytes)=204795904
Peak Reduce Virtual memory (bytes)=2821476352
Shuffle Errors
BAD_ID=0
CONNECTION=0
IO_ERROR=0
WRONG_LENGTH=0
WRONG_MAP=0
WRONG_REDUCE=0
File Input Format Counters
Bytes Read=1128
File Output Format Counters
Bytes Written=429
finished!
holmes@holmes-Lenovo-XiaoXin-Air-14IKBR:/usr/local/hadoop$


新标签页
All Applications
Browsing HDFS
localhost:9870/explorer.html#/user/holmes/output
Hadoop
Overview
Datanodes
Datanode Volume Failures
Snapshot
Startup Progress
Utilities
Browse Directory
/user/holmes/output
Go!
Show 25 entries
Search:
Permission
Owner
Group
Size
Last Modified
Replication
Block Size
Name
-rw-r--r--
holmes
supergroup
0 B
Nov 14 20:27
1
128 MB
_SUCCESS
-rw-r--r--
holmes
supergroup
429 B
Nov 14 20:27
1
128 MB
part-r-00000
Showing 1 to 2 of 2 entries
Previous
1
Next
Hadoop, 2021.


新标签页
All Applications
localhost:8088/cluster
All Applications
Cluster
About
Nodes
Node Labels
Applications
NEW
NEW_SAVING
SUBMITTED
ACCEPTED
RUNNING
FINISHED
FAILED
KILLED
Scheduler
Tools
Cluster Metrics
Apps Submitted
Apps Pending
Apps Running
Apps Completed
Containers Running
Used Resources
Total Resources
Reserved Resources
1
0
0
1
0
<memory:0, vCores:0>
<memory:8192, vCores:8>
<memory:0, vCores:0>
76
Cluster Nodes Metrics
Active Nodes
Decommissioning Nodes
Decommissioned Nodes
Lost Nodes
Unhealthy Nodes
Rebooted Node
1
0
0
0
0
0
Scheduler Metrics
Scheduler Type
Scheduling Resource Type
Minimum Allocation
Maximum Allocation
Capacity Scheduler
[memory-mb (unit=Mi), vcores]
<memory:1024, vCores:1>
<memory:8192, vCores:4>
0
Show 20 entries
ID
User
Name
Application Type
Queue
Application Priority
StartTime
LaunchTime
FinishTime
State
FinalStatus
Running Containers
Allocated CPU VCores
Allocated Memory MB
Allocated GPUs
Reserved CPU VCores
Reserved Memory MB
Reserved GPUs
application_1636892254552_0001
holmes
kNNJob
MAPREDUCE
default
0
Sun Nov 14 20:27:24 +0800 2021
Sun Nov 14 20:27:25 +0800 2021
Sun Nov 14 20:27:38 +0800 2021
FINISHED
SUCCEEDED
N/A
N/A
N/A
N/A
N/A
N/A
N/A
Showing 1 to 1 of 1 entries


新标签页
All Applications
Browsing HDFS
localhost:9870/explorer.html#/user/holmes/output
Hadoop
Overview
Datanodes
Snapshot
Startup Progress
Utilities
Browse Director
/user/holmes/output
Go!
Show 25 entries
Permission
Owner
Name
-rw-r--r--
_SUCCESS
-rw-r--r--
part-r-00000
Showing 1 to 2 of 2 entries
Previous
1
Next
Hadoop, 2021.
File information - part-r-00000
Download
Head the file (first 32K)
Tail the file (last 32K)
Block information --
Block 0
Block ID: 1073742147
Block Pool ID: BP-166325193-127.0.1.1-1633417862071
Generation Stamp: 1323
Size: 429
Availability:
holmes-Lenovo-XiaoXin-Air-14IKBR
File contents
setosa
versicolor
versicolor
setosa
versicolor
virginica
versicolor
setosa
Close