《大数据开发技术（Hadoop）》

实训报告

主题：自选主题实训项目

组长：杜春梅（22351013112）

成员：周学政（22351013114）

梁倚凡（22351013110）

完成日期：2024.12.31

目 录

[第1部分 项目目的 1](#_Toc1057)

[第2部分 项目实现技术 1](#_Toc27978)

[第3部分 项目实现思路介绍 2](#_Toc20657)

[第4部分 项目实现流程介绍 2](#_Toc22979)

[第5部分 项目分析结果说明 4](#_Toc13545)

[第6部分 项目代码介绍文档 5](#_Toc1485)

# 项目目的

# 项目目的如下：

1、掌握 Hadoop 分布式集群环境的搭建和配置。

2、学会利用多种工具（如 Spark、Python）

对大数据进行清洗、存储、分析与处理。

3、学会利用多种可视化工具

完成数据分析结果的可视化展示。

4、培养学生的项目开发能力，熟悉大数据项目开发流程

及版本管理工具 （GitHub）的使用。

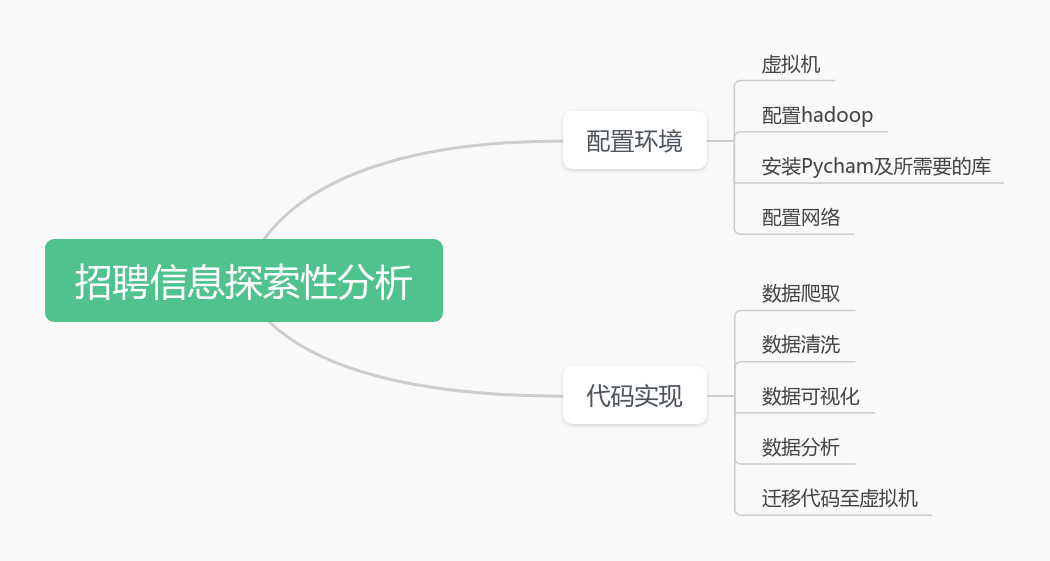
# 项目实现技术

项目基于hadoop技术进行开发，开发及运行的软件环境为：

* Vmware虚拟机
* Hadoop3.3.5
* Spark-2.1.0
* Python-3.7
* Annaconda-5.3.1
* Seaborn-0.9.0

# 项目实现思路介绍

3.1整体实现思路与实现步骤流程图



# 项目实现流程介绍

**4.1搭建完全分布式Hadoop集群部分**

步骤一：选定一台机器作为 Master；

步骤二：在Master节点上创建hadoop用户、安装SSH服务端、安装Java环境，安装Hadoop，并完成伪分布配置；

步骤三：对Master节点进行与外网连接的配置，使其能ping到本地以及到外网（如www.baidu.com）；

步骤三：创建新的虚拟机，称为Slave1，在Slave1节点上创建hadoop用户、安装SSH服务端、安装Java环境、连接外网；

步骤四：通过网络配置将Master和Slave1连接起来，使其相互ping通，将Master节点上的“/usr/local/hadoop”目录复制到Slave1节点上；在Master节点上开启Hadoop；

步骤五：通过对完成配置的Slave1节点复制克隆出另一个Slave节点，称为Slave2，接着对这一个主节点和两个从节点进行网络配置，重复步骤四；

步骤六：安装Pycharm软件，安装annaconda，以及各种库如numpy、pandas、matplotlib、jieba等，并安装MYSQL数据库系统、pySpark接口等。

**4.2数据采集与存储部分**

在招聘数据分析中，数据从章鱼公有云平台“对招聘网站分析”的实训板块获取。搜集到的数据包括职位名称、公司名称、薪资范围、工作地点、学历要求、工作经验等。

**4.3数据清洗**

数据清洗与处理是确保数据准确性和一致性的关键步骤，包括去除重复数据、处理缺失值、数据格式转换等。

首先，对于存在空缺值的数据，我们进行整行数据删除。

对于公司规模数据，我们通过value.counts( )发现数据存在数据区间不一致，数据划分较为散乱的情况。因此，我们将数据进行清洗，结合value.counts( )反映出数据量以及临界值，将公司规模数据统一划分并归类为七个区间，分别是“保密”“20人以下”“20-99人”“100-499人”“500-999人”“1000-9999人”“10000人以上”。

对于工作地点数据，我们通过value.counts( )发现，有些数据细化到区，而有些只细化到市，因此，我们将数据进行清洗，利用split( )方法，将工作地点数据整理为只细化到市，分别是“北京”“上海”“深圳”“广州”。

对于月工资数据，我们通过value.counts( )发现数据也存在数据区间不一致，数据划分较为散乱的情况。因此，在可视化阶段，我们将数据进行简单划分和归类，结合value.counts( )反映出数据量以及临界值，将数据分为八个区间，分别是“面议”“6000元/月以下”“6000-8000元/月”“8000-10000元/月”“10000-20000元/月”“20000-30000元/月”“30000-50000元/月”“500000元/月以上”。在数据分析阶段，为了预测薪资范围，我们目标是将数据清洗数据类型为列表，[XXX,XXX]区间形式的数据，通过查看value.counts( )，我们将月工资为“面议”的数据进行删除，对薪资为“XX以下”的数据整理成[0,XXX]，对于其他数据，利用split( )和apply(lambda x: )方法，将数据整理成[XXX,XXX]的区间形式，方便后续的模型预测。

**4.4数据分析**

在本次招聘数据可视化中，为了发现数据中的模式、趋势和关联，我们使用清洗后的数据，通过Matplotlib、wordcloud、Seaborn等库绘制柱状图、堆叠柱状图、饼状图、词云图等展示职位需求、公司规模、行业分布、公司性质、薪资分布、学历影响、经验分布、工作地点等。

对于本次招聘数据，我们选择的探索方向是利用公司规模、公司性质、经验等作为特征，对薪资区间进行预测。首先利用LabelEncoder（）与独热编码对特征进行处理，合并特征，然后切分训练集与测试集，放入随机森林模型中进行训练，随后进行预测，查看预测值与真实值的差别，得到较好的薪资预测模型。

# 项目分析结果说明

**5.1需求处理与项目理解**

作为数据分析项目的起始阶段，明确分析目标和需求背景。在招聘数据分析中，这包括理解招聘市场的趋势、特定行业或职位的招聘需求以及薪资预测等。数据处理的重点是确认业务目标、数据指标和核心维度，确保分析工作有明确的方向和目标。在本次实训中，我们目标明确、分工合作，在短时间内达到较好的实训效果。

**5.2数据收集与处理**

高质量的数据是进行准确分析的前提。如果数据中包含错误或偏差，分析结果可能会误导决策。在查看公有云平台的数据后，通过数据清洗有助于消除错误数据和偏差，确保分析结果的可靠性。我们发现采集到的数据存在缺失值以及数值类型不一致的数据，通过数据清洗去除数据中的错误、重复、不完整或无关的信息，从而提高数据的质量。通过清洗，数据变得更加整洁和一致，这使得数据更容易被独热编码模型以及随机森林模型所使用。同时，清洗后的数据能够更准确地反映现实情况，从而支持更明智的决策制定。

**5.3数据分析**

通过数据可视化，我们绘制了“公司规模分布情况”“公司性质分布情况”“经验分布情况”“公司行业分布情况”“职位类别分布情况”“工作地点分布情况”“月工资分布情况”“各学历等级的工资分布情况”“工作地点的工资分布情况”“各经验等级的工资分布情况”等柱状图、堆叠柱状图、饼状图、词云图，直观地展示了数据。

从图上可以看出，公司规模主要为100-499人，民营较多，以互联网产业为主，经验要求不限较多，招聘需求最多的岗位是数据库开发工程师和高级软件工程师，工作地点主要分布在北京，薪资范围主要是6000-8000元/月，其中，在各薪资范围中，学历要求为本科生占大部分。

通过对特征进行特征化处理以及随机森林模型训练，可以根据公司规模、公司性质、经验等特征，对薪资有较准确的预测，对求职者职业规划提供有效数据支持。

# 项目代码介绍文档

**6.1搭建完全分布式Hadoop集群部分**

### 创建虚拟机Master节点，以及对Master节点进行配置

·这些步骤都是在终端进行输入

### 6.1.1.1创建hadoop用户

|  |
| --- |
| sudo useradd -m hadoop -s /bin/bash # 创建了可以登陆的hadoop用户  sudo passwd hadoop # 设置密码  sudo adduser hadoop sudo # 为hadoop用户增加管理员权限 |

### 6.1.1.2 安装SSH服务端

|  |
| --- |
| sudo apt-get update # 更新apt  sudo apt-get install vim # 下载vim配置文件  sudo apt-get install openssh-server # 安装SSH server  ssh localhost # 登陆本机，首次登录需输入hadoop密码 |

接下来可以配置SSH无密码登录

|  |
| --- |
| exit # 退出刚才的 ssh localhost  cd ~/.ssh/  ssh-keygen -t rsa # 会有提示，都按回车就可以  cat ./id\_rsa.pub >> ./authorized\_keys # 加入授权 |

此时再用 ssh localhost 命令，无需输入密码就可以直接登陆。

### 6.1.1.3 安装Java环境

首先通过VMware Tools软件将JDK安装包jdk-8u371-linux-x64.tar.gz上传到~/Downloads目录下

|  |
| --- |
| cd /usr/lib  sudo mkdir jvm #创建/usr/lib/jvm目录用来存放JDK文件  cd ~ #进入hadoop用户的主目录  cd Downloads  sudo tar -zxvf ./jdk-8u371-linux-x64.tar.gz -C /usr/lib/jvm #把JDK文件解压到/usr/lib/jvm目录下 |

将JDK文件解压缩后，可以执行如下命令到/usr/lib/jvm目录查看一下：

|  |
| --- |
| cd /usr/lib/jvm  ls |



用vim编辑器在.bashrc文件添加以下内容，设置环境变量：

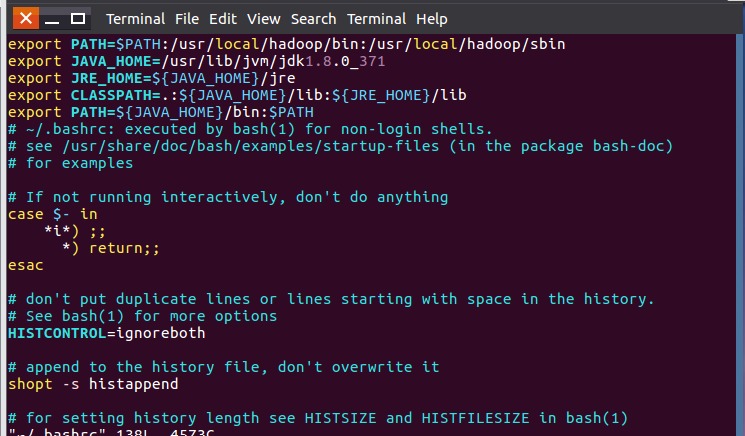
export JAVA\_HOME=/usr/lib/jvm/jdk1.8.0\_371

export JRE\_HOME=${JAVA\_HOME}/jre

export CLASSPATH=.:${JAVA\_HOME}/lib:${JRE\_HOME}/lib

export PATH=${JAVA\_HOME}/bin:$PATH

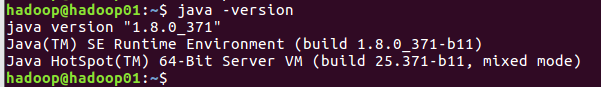
|  |
| --- |
| cd ~  vim ~/.bashrc |

保 存.bashrc文件并退出vim编辑器。然后，继续执行如下命令让.bashrc文件的配置立即生效：

|  |
| --- |
| source ~/.bashrc |

可使用如下命令查看是否安装成功：

|  |
| --- |
| java -version |

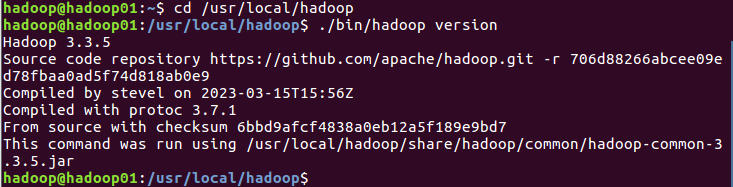


### 6.1.1.4 安装Hadoop3.3.5并进行伪分布式配置

|  |
| --- |
| sudo tar -zxvf ~/下载/hadoop-3.3.5.tar.gz -C /usr/local # 解压到/usr/local中  cd /usr/local/  sudo mv ./hadoop-3.3.5/ ./hadoop # 将文件夹名改为hadoop  sudo chown -R hadoop ./hadoop # 修改文件权限 |

Hadoop解压后即可使用，输入以下命令查看Hadoop是否可以，成功则显示Hadoop版本信息：

|  |
| --- |
| cd /usr/local/hadoop  ./bin/hadoop version |



接着可进行Hadoop伪分布式配置：

|  |
| --- |
| cd /usr/local/hadoop/etc/hadoop/ # 进入配置文件的目录 |

1. **修改配置文件 core-site.xml**

|  |
| --- |
| vim core-site.xml # 进入配置文件的目录 |

将其中的：

|  |
| --- |
| <configuration>  </configuration> |

修改为下面的配置：

|  |
| --- |
| <configuration>  <property>  <name>hadoop.tmp.dir</name>  <value>file:/usr/local/hadoop/tmp</value>  <description>Abase for other temporary directories.</description>  </property>  <property>  <name>fs.defaultFS</name>  <value>hdfs://localhost:9000</value>  </property>  </configuration> |

1. **修改配置文件 hdfs-site.xml**

|  |
| --- |
| vim hdfs-site.xml# 进入配置文件的目录 |

将其中的：

|  |
| --- |
| <configuration>  </configuration> |

修改为下面的配置：

|  |
| --- |
| <configuration>  <property>  <name>dfs.replication</name>  <value>1</value>  </property>  <property>  <name>dfs.namenode.name.dir</name>  <value>file:/usr/local/hadoop/tmp/dfs/name</value>  </property>  <property>  <name>dfs.datanode.data.dir</name>  <value>file:/usr/local/hadoop/tmp/dfs/data</value>  </property>  </configuration> |

1. **执行NameNode格式化**

|  |
| --- |
| cd /usr/local/hadoop  ./bin/hdfs namenode -format |

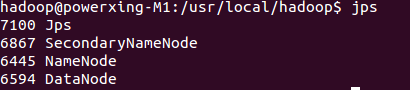
成功后，可以看到“successfully formatted”的提示。

1. **开启 NameNode 和 DataNode 守护进程**

|  |
| --- |
| cd /usr/local/hadoop  ./sbin/start-dfs.sh #start-dfs.sh是个完整的可执行文件，中间没有空格 |

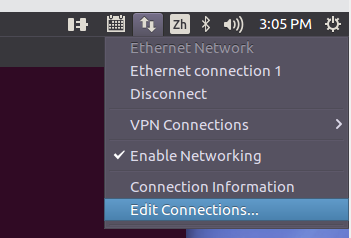
1. **启动完成后，可以通过命令 jps 来判断是否成功启动**

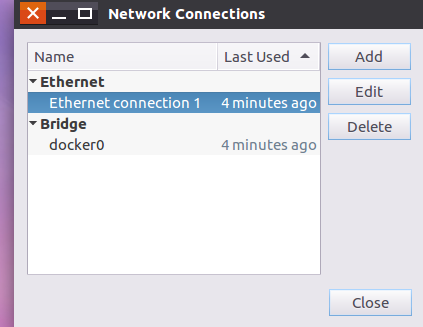
如果成功启动则会列出如下进程: "NameNode"、"DataNode" 和 "SecondaryNameNode"



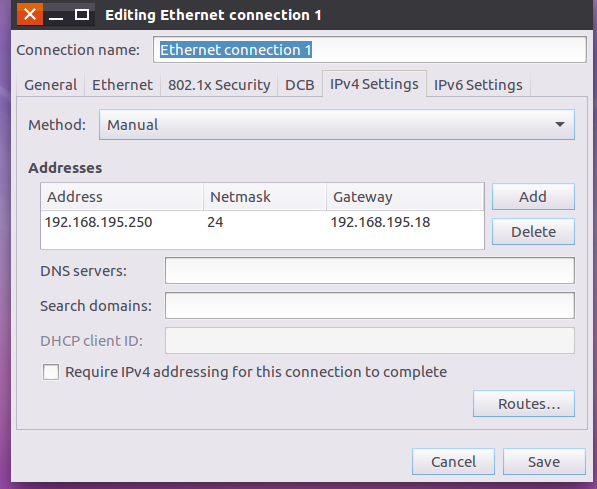
### 6.1.1.5 进行无线网络配置

1. Linux系统配置



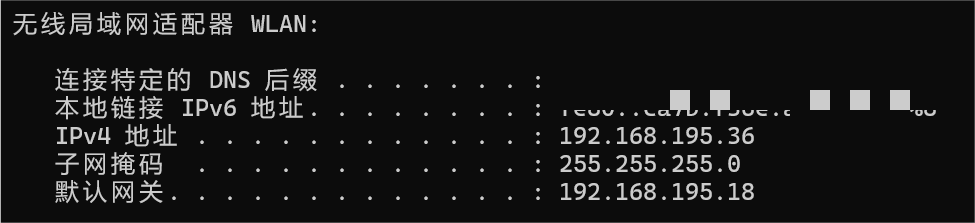


选择Edit



1. 选择IPv4 Settings
2. 选择Method：Manual
3. 选择 Add
4. 分别填入Address、Netmask、Gateway
5. Address:需要参考主机Windows系统的ip4 地址，可通过ipconfig命令查询；

对宿主机即本地机使用“win+r”-> 输入“cmd” 回车，打开wind命令行；输入ipconfig命令：



其中Address要求前三位与IPv4前三位一致，最后一位则要求在[36, 255]内，选择将Master的IP地址写为：192.168.195.250；

1. Netmask:可直接设置255.255.255.0； 下图中是设置之后的效果，数值会发生变化，忽略即可；
2. Gateway:可直接参考主机windows系统中的网关设置，可通过ipconfig命令查询；

修改完成后选择save保存。

1. VMware配置



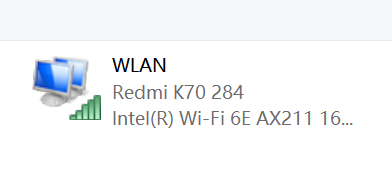
1. 选择虚拟网络编辑器
2. 选择更改设置



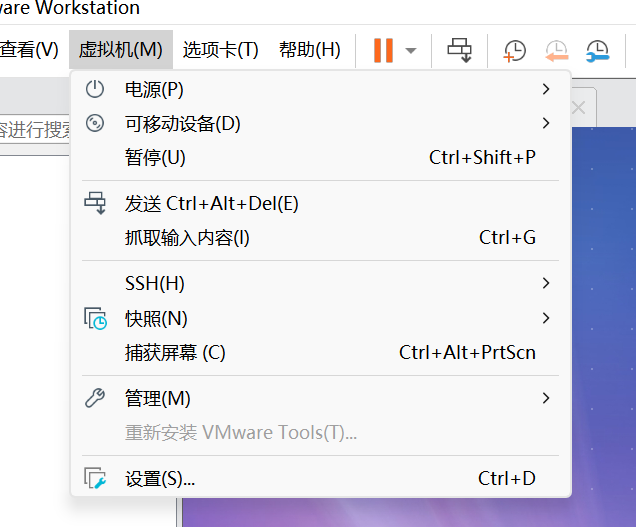
1. 点击名称为VMnet0
2. 在VMnet信息选择桥接模式
3. 已连接中选择Intel(R) Wi-Fi 6E AX211 160MHz

该步骤需需要查看主机windows中使用的无线网网适配器名称

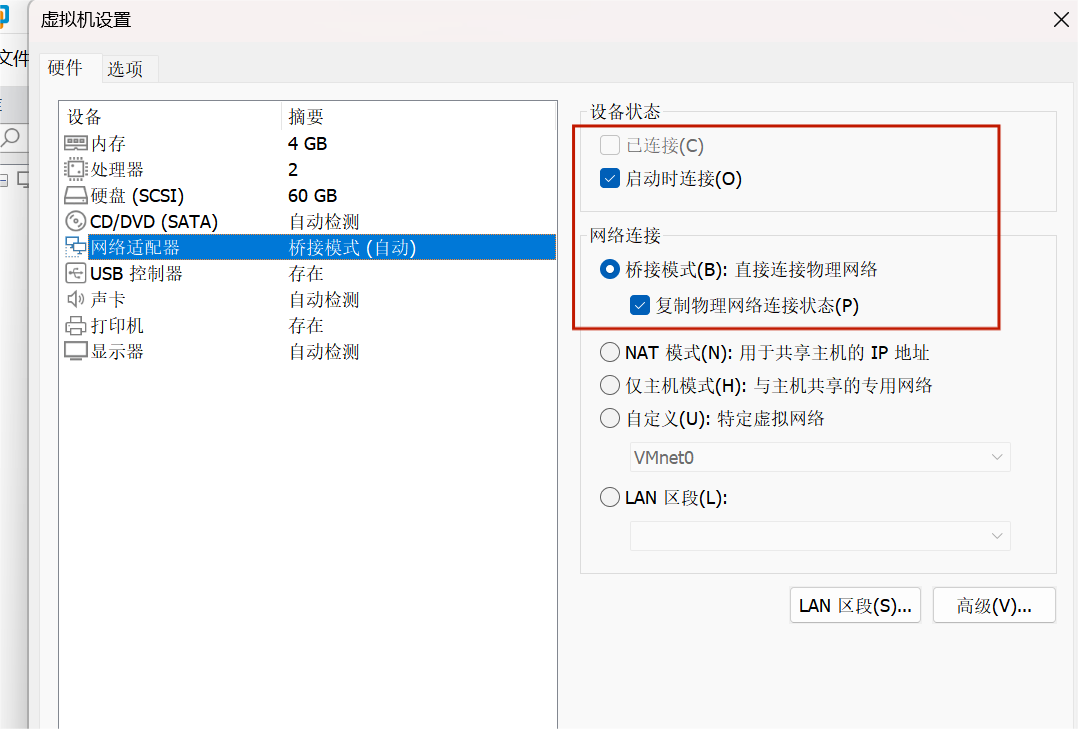
操作过程：控制面板->网络和Internet->网络和共享中心->更改适配器设置



1. 修改完成后，点击应用 -> 确认

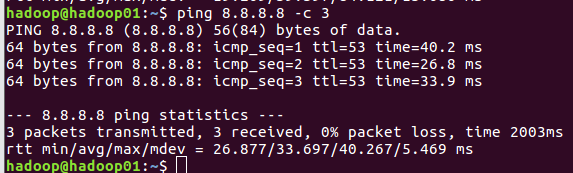


1. 点击虚拟机->设置->网络适配器



1. 选择桥接模式->将红框中的选项勾选上->点击确定
2. 与外网连接

上述内容配置完成后，输入ping 8.8.8.8，是可以ping通

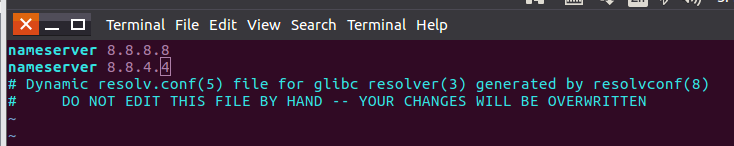


但要与外网连接需修改resolv.conf文件

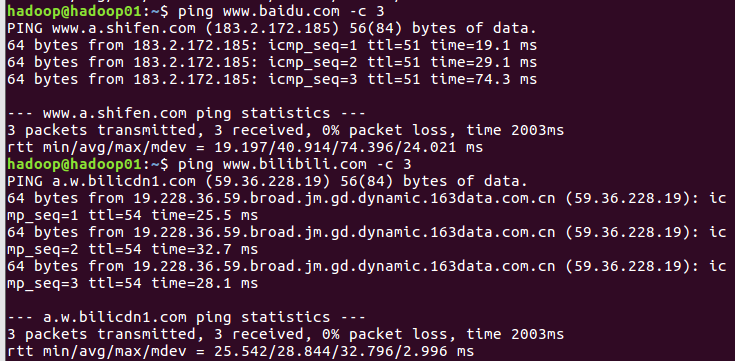
|  |
| --- |
| sudo vim /etc/resolv.conf |

添加如下内容：

|  |
| --- |
| nameserver 8.8.8.8  nameserver 8.8.4.4 |



完成上述操作便可以ping通任意网站了



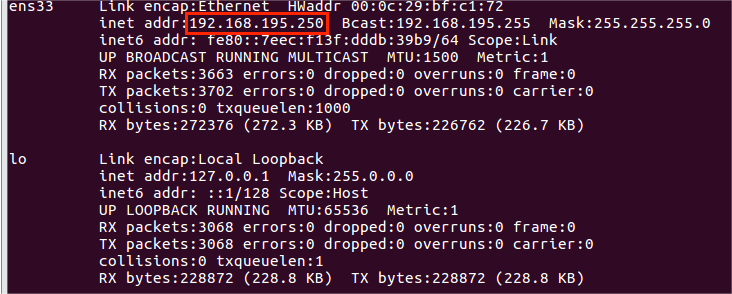
### 创建虚拟机Slave1节点并完成配置

**6.1.2.1 在Slave1节点上创建hadoop用户、安装SSH服务端、安装Java环境、连接外网。**

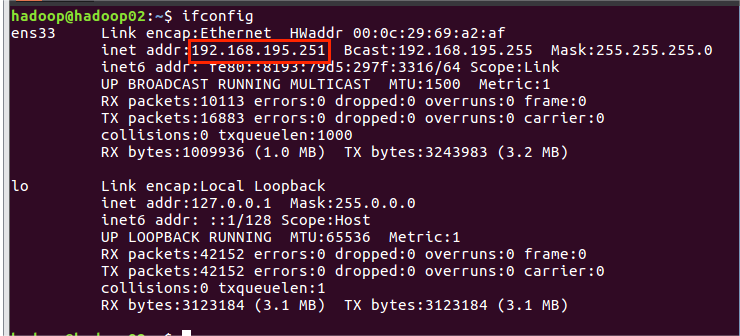
Slave节点操作与Master节点操作基本一致，只需进行到6.1.1.3步骤即可。由于两台虚拟机之间要连接即ping通，需处于一个网络设置环境中，所以也要对Slave1节点进行网络配置，即6.1.1.5步骤也需进行。但是对于Slave1节点的Address不能与Master节点的Address一致，即IP地址不能一致，最后一位IP地址在区间[36,255]选择除250以外的数字，在此将Slave1节点的IP地址修改为：192.168.195.251

则修改后，对Master节点和Slave1节点这两个的终端分别输入ifconfig，得到：

Master节点



Slave1节点

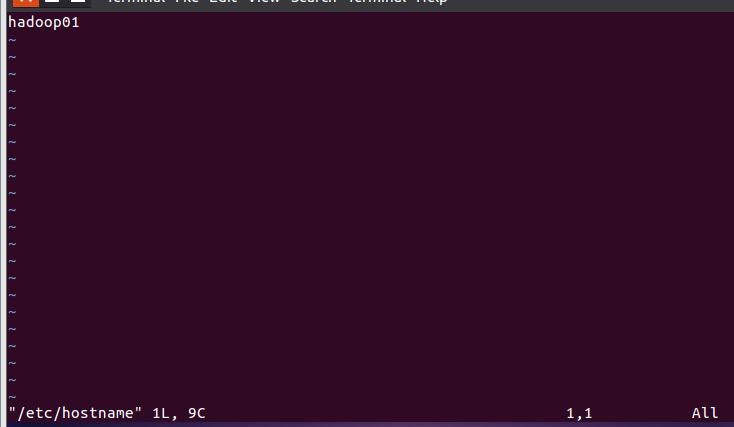


**6.1.3 完成一个主节点和从节点的连接**

**6.1.3.1 网络配置**

为了便于区分Master节点和Slave节点，可以修改各个节点的主机名：

|  |
| --- |
| sudo vim /etc/hostname |



将Master节点的主机名修改为：hadoop01

将Slave1节点的主机名修改为：hadoop02

其他节点一样

修改完成后，都需要重启，重启后的Shell命令提示符状态就会显示修改后的主机名



由6.1.2可知，通过ifconfig得出两个主机的IP地址分别为：

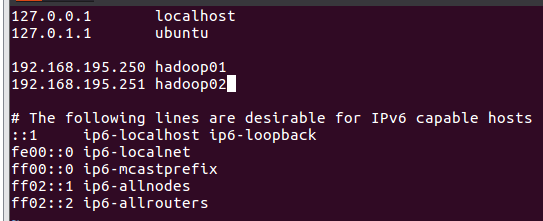
192.168.195.250 hadoop01

192.168.195.251 hadoop02

然后修改Master节点中的“/etc/hosts”文件：

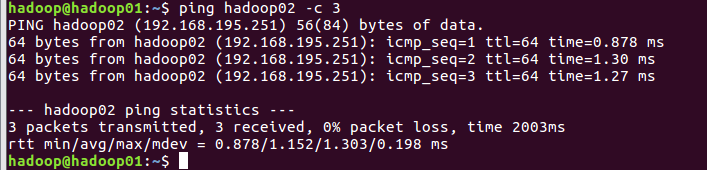
|  |
| --- |
| sudo vim /etc/hosts |

输入以上两条IP和主机名映射关系：

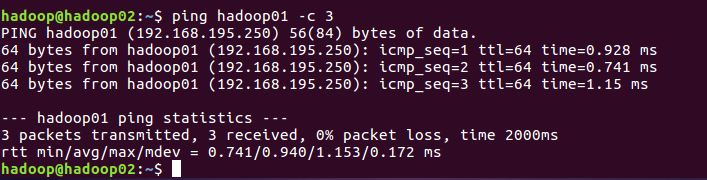


保存修改后重启系统，同时Slave1节点也是一样的操作。

Master节点和Slave1节点重启后，可在Master节点输入ping hadoop02 -c 3：



在Slave节点输入ping hadoop01 -c 3



**6.1.3.2 SSH无密码登录**

首先在hadoop02执行如下命令

|  |
| --- |
| sudo apt-get install openssh-server # 按照你装SSH服务器，如已安装就不用重复安装 |

然后，生成Master节点（hadoop01）的公匙，如果之前已经生成过公钥（在3.3.3节安装伪分布式模式的Hadoop时生成过一次公钥），必须要删除原来生成的公钥，重新生成一次，因为前面我们对主机名进行了修改。在Master节点执行如下命令：

|  |
| --- |
| cd ~/.ssh # 如果没有该目录，先执行一次ssh localhost  rm ./id\_rsa\* # 删除之前生成的公匙（如果已经存在）  ssh-keygen -t rsa # 执行该命令后，遇到提示信息，一直按回车就可以 |

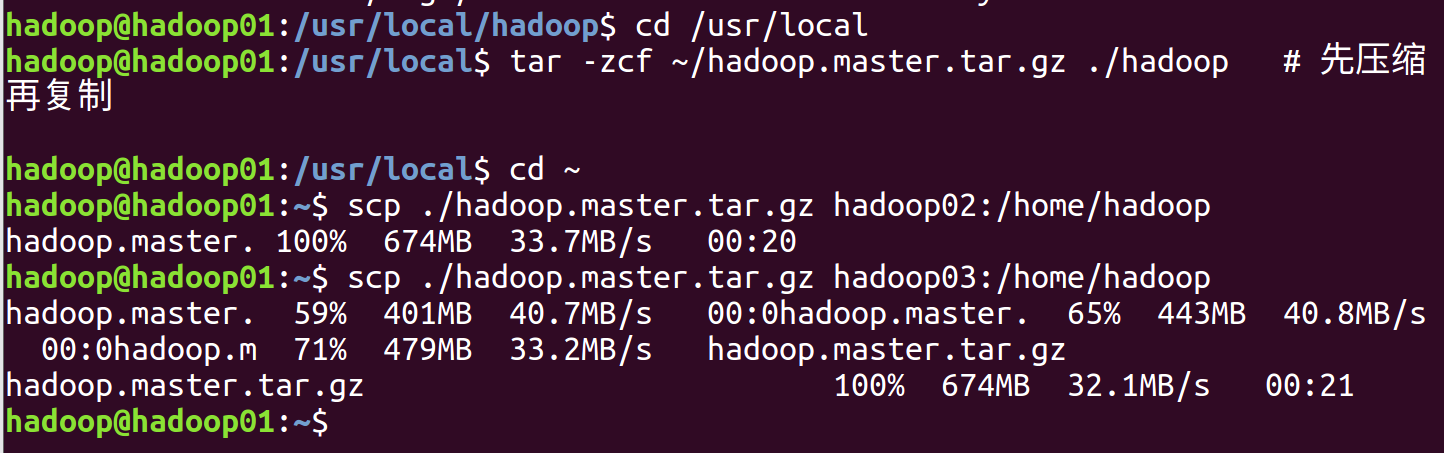
为了让Master节点能够无密码SSH登录本机，需要在Master节点上执行如下命令：

|  |
| --- |
| cat ./id\_rsa.pub >> ./authorized\_keys |

完成后可以执行命令“ssh hadoop01”来验证一下，可能会遇到提示信息，只要输入yes即可，测试成功后，请执行“exit”命令返回原来的终端。

接下来，在Master节点（hadoop01）将上公匙传输到Slave节点（hadoop02）：

|  |
| --- |
| scp ~/.ssh/id\_rsa.pub hadoop@hadoop02:/home/hadoop/ |



传输完成以后，在hadoop02上的“/home/hadoop”目录下就可以看到文件id\_rsa.pub了。

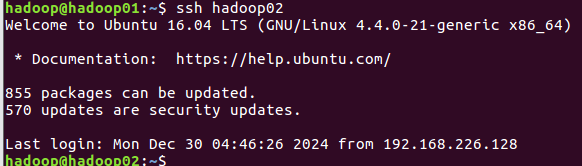
接着在Slave节点（hadoop02）上执行如下命令将SSH公匙加入授权：

|  |
| --- |
| mkdir ~/.ssh # 如果不存在该文件夹需先创建，若已存在，则忽略本命令  cat ~/id\_rsa.pub >> ~/.ssh/authorized\_keys  rm ~/id\_rsa.pub # 用完以后就可以删掉 |

如果有其他Slave节点，也要执行将Master公匙传输到Slave节点以及在Slave节点上加入授权这两步操作。

这样，在Master节点上就可以无密码SSH登录到各个Slave节点了，可在Master节点（hadoop01）上执行如下命令进行检验：

|  |
| --- |
| ssh hadoop02 |



**6.1.3.3 配置PATH变量**

在Master节点上进行配置：

|  |
| --- |
| vim ~/.bashrc |

添加以下内容：

export PATH=$PATH:/usr/local/hadoop/bin:/usr/local/hadoop/sbin

保存后执行命令使其生效：

|  |
| --- |
| source ~/.bashrc |

**6.1.3.4 配置集群/分布式环境**

在配置集群/分布式模式时，需要修改“/usr/local/hadoop/etc/hadoop”目录下的配置文件，这里仅设置正常启动所必须的设置项，包括workers、core-site.xml、hdfs-site.xml、mapred-site.xml、yarn-site.xml共5个文件。

可参考该地址去修改<https://dblab.xmu.edu.cn/blog/4202/>

**6.1.3.5 启动环境**

上述5个文件全部配置完成以后，需要把Master节点上的“/usr/local/hadoop”文件夹复制到各个节点上。如果之前已经运行过伪分布式模式，建议在切换到集群模式之前首先删除之前在伪分布式模式下生成的临时文件。具体来说，需要首先在Master节点上执行如下命令：

|  |
| --- |
| cd /usr/local/hadoop  sudo rm -r ./tmp # 删除 Hadoop 临时文件  sudo rm -r ./logs/\* # 删除日志文件  cd /usr/local  tar -zcf ~/hadoop.master.tar.gz ./hadoop # 先压缩再复制  cd ~  scp ./hadoop.master.tar.gz hadoop02:/home/hadoop |

然后在hadoop02节点上执行如下命令：

|  |
| --- |
| cd ~  sudo rm -r /usr/local/hadoop # 删掉旧的（如果存在）  sudo tar -zxf ~/hadoop.master.tar.gz -C /usr/local  sudo chown -R hadoop /usr/local/hadoop |

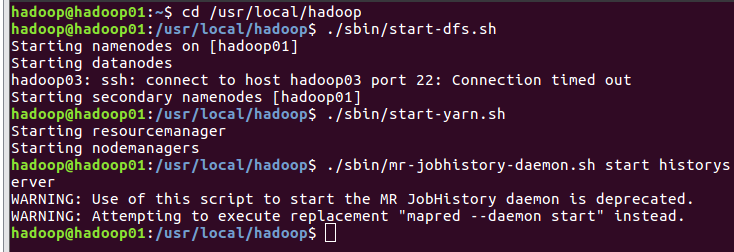
同样，如果有其他Slave节点，也要执行将hadoop.master.tar.gz传输到Slave节点以及在Slave节点解压文件的操作。

首次启动Hadoop集群时，需要先在Master节点（hadoop01）执行名称节点的格式化（只需要执行这一次，后面再启动Hadoop时，不要再次格式化名称节点），命令如下：

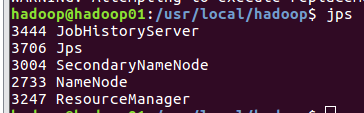
|  |
| --- |
| cd /usr/local/hadoop  ./bin/hdfs namenode -format |

现在就可以启动Hadoop了，启动需要在Master节点（hadoop01）上进行，执行如下命令：

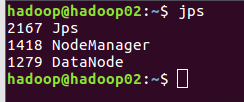
|  |
| --- |
| cd /usr/local/hadoop  ./sbin/start-dfs.sh  ./sbin/start-yarn.sh  ./sbin/mr-jobhistory-daemon.sh start historyserver |



通过命令jps可以查看各个节点所启动的进程。如果已经正确启动，则在Master节点上可以看到NameNode、ResourceManager、SecondaryNameNode和JobHistoryServer进程

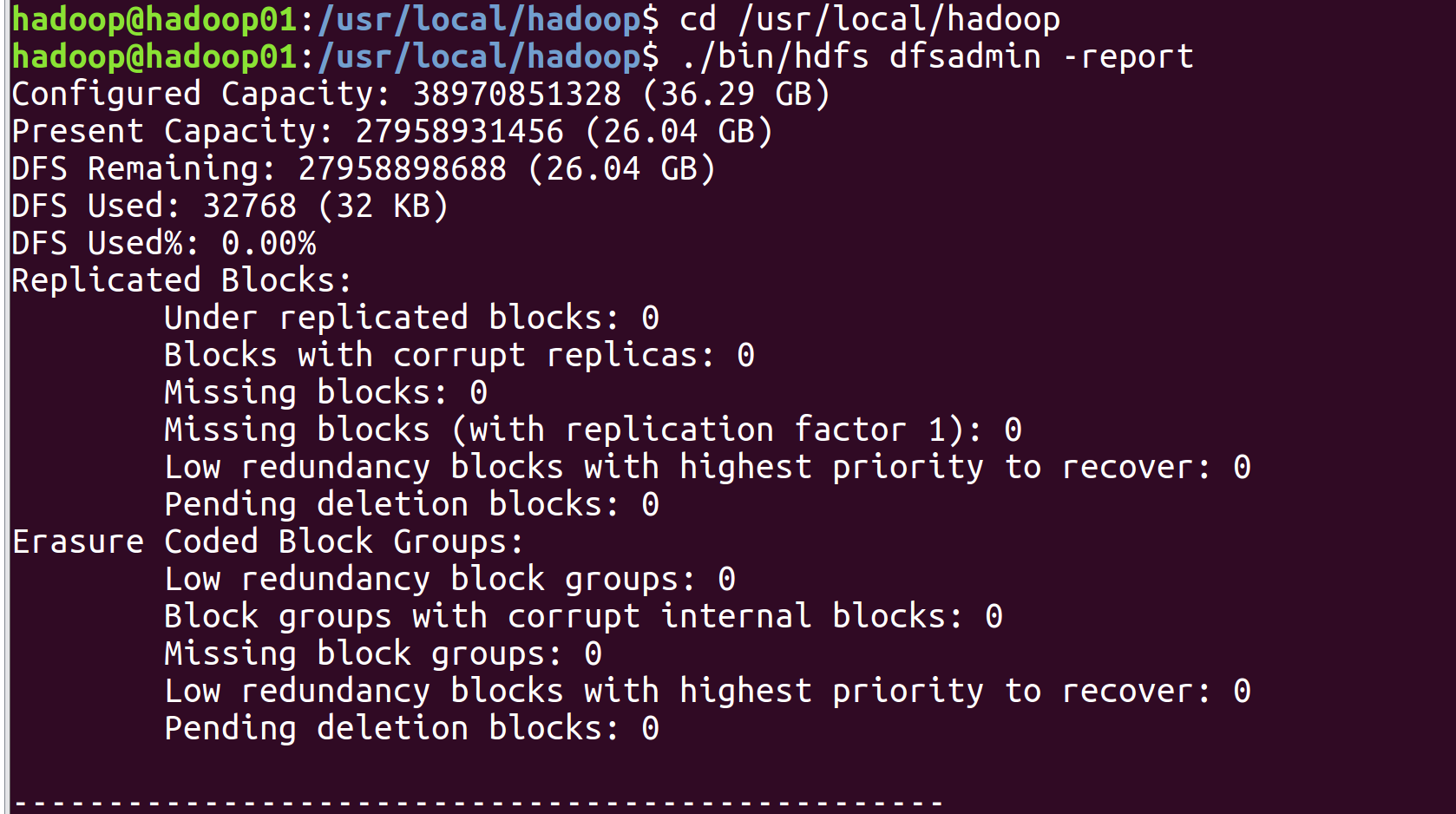


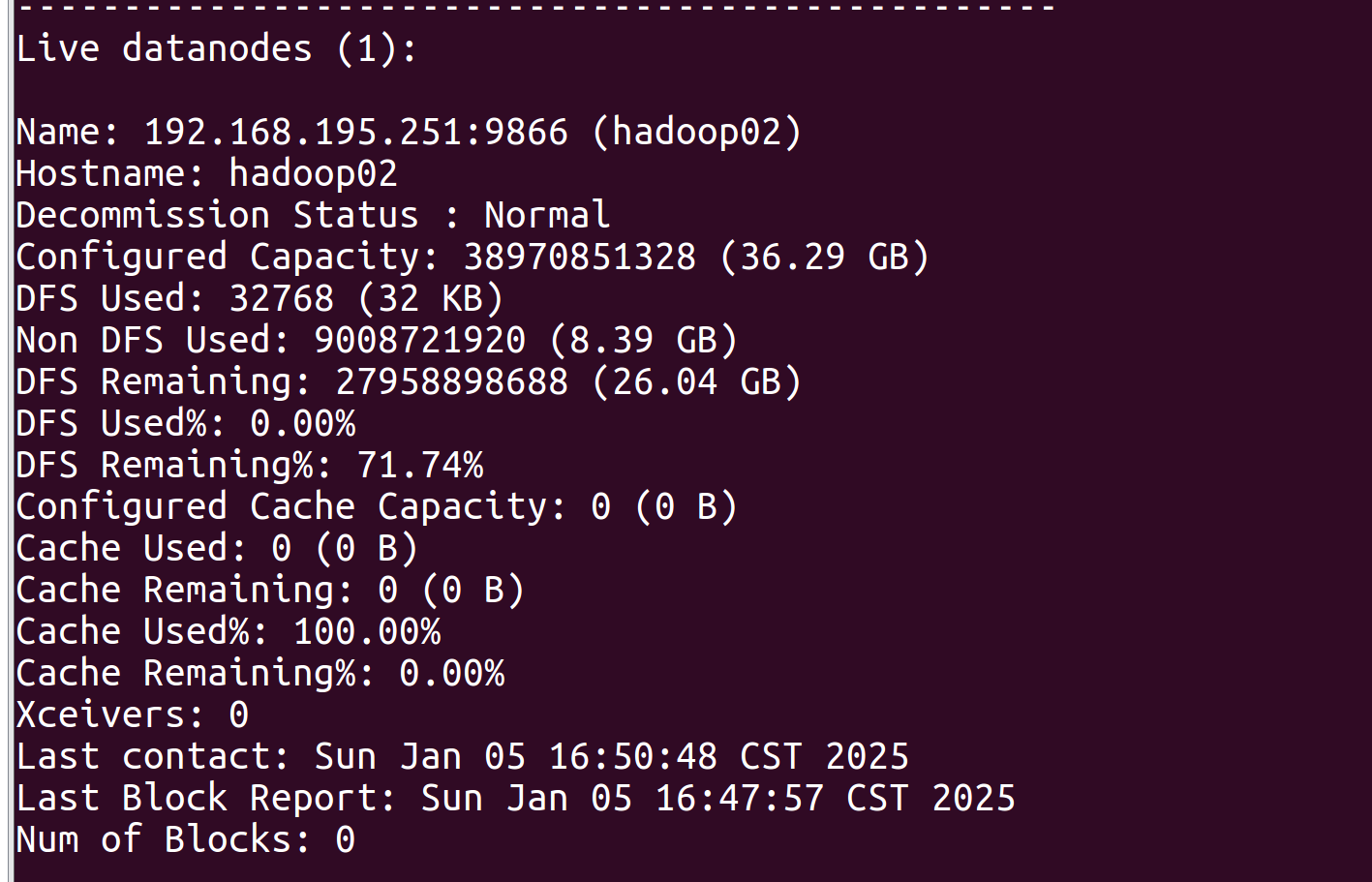
在Slave节点可以看到DataNode和NodeManager进程



缺少任一进程都表示出错。另外还需要在Master节点上通过如下命令查看数据节点是否正常启动：

|  |
| --- |
| cd /usr/local/hadoop  ./bin/hdfs dfsadmin -report |





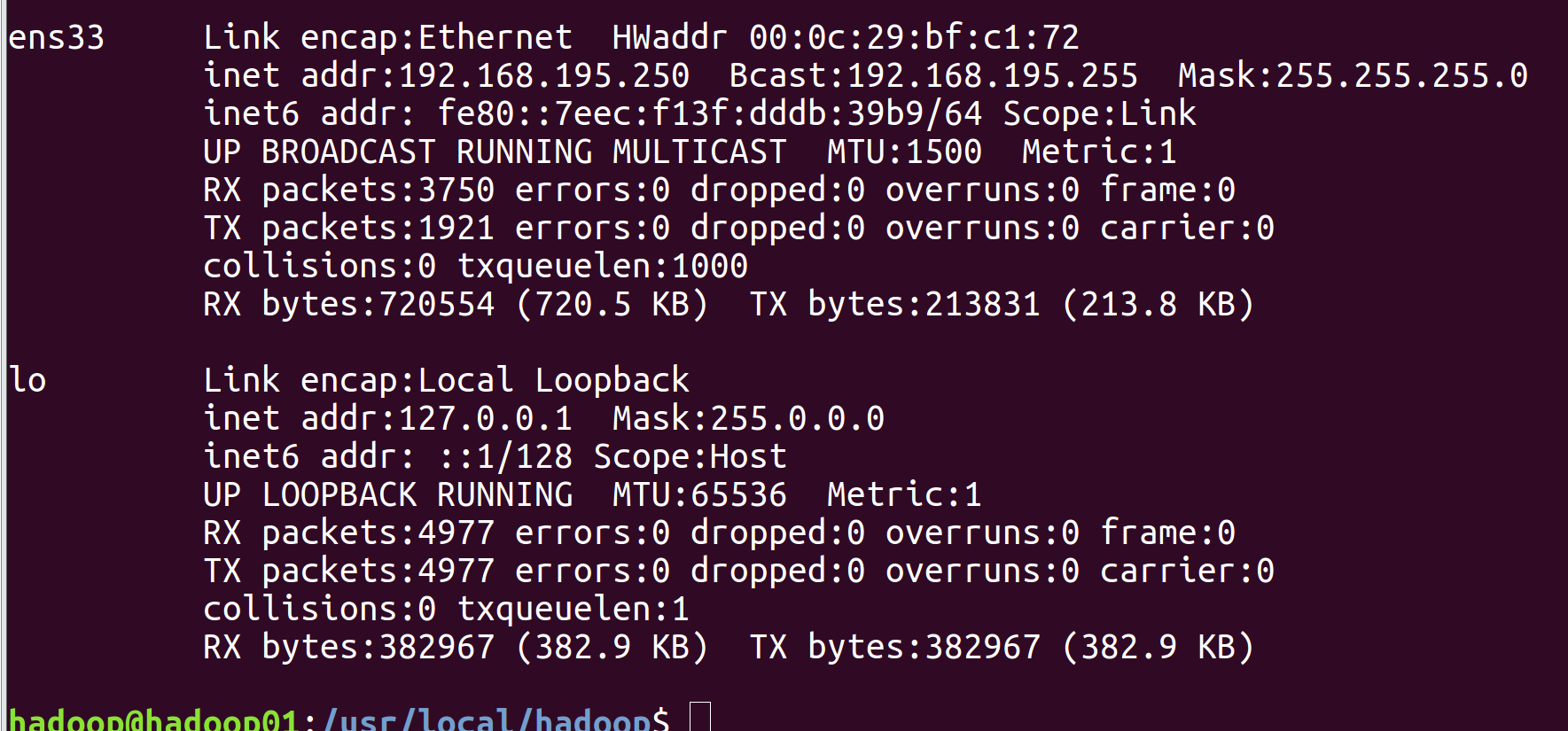
至此一个主节点和一个从节点已配置完成。

**6.1.3.6 将Slave1复制克隆成Slave2,完成一个主节点和两个从节点的配置**

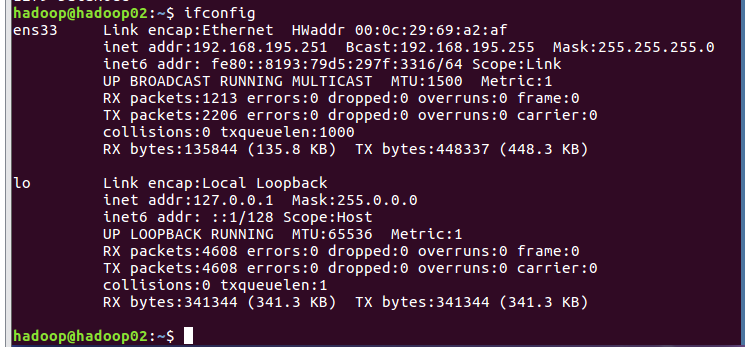
1. 将Slave2的主机名改为hadoop03
2. 修改Slave2的IP地址即Address为：192.168.195.252

修改后这三个节点的IP地址分别为:

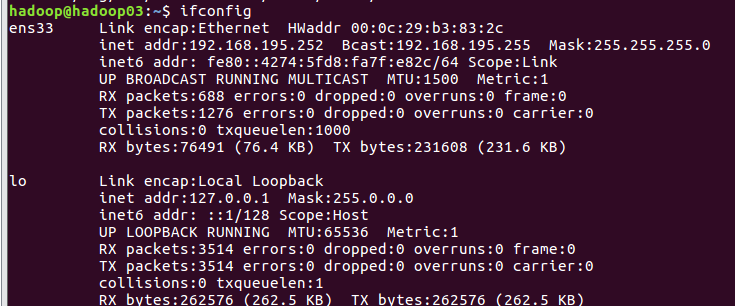
Master



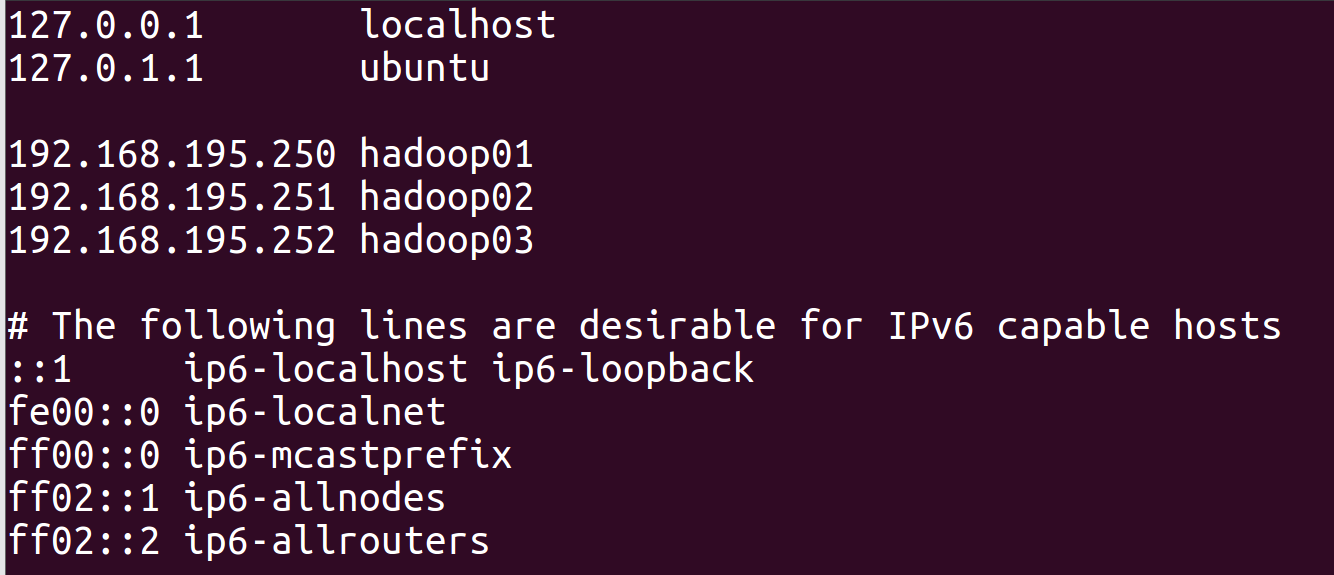
Slave1



Slave2

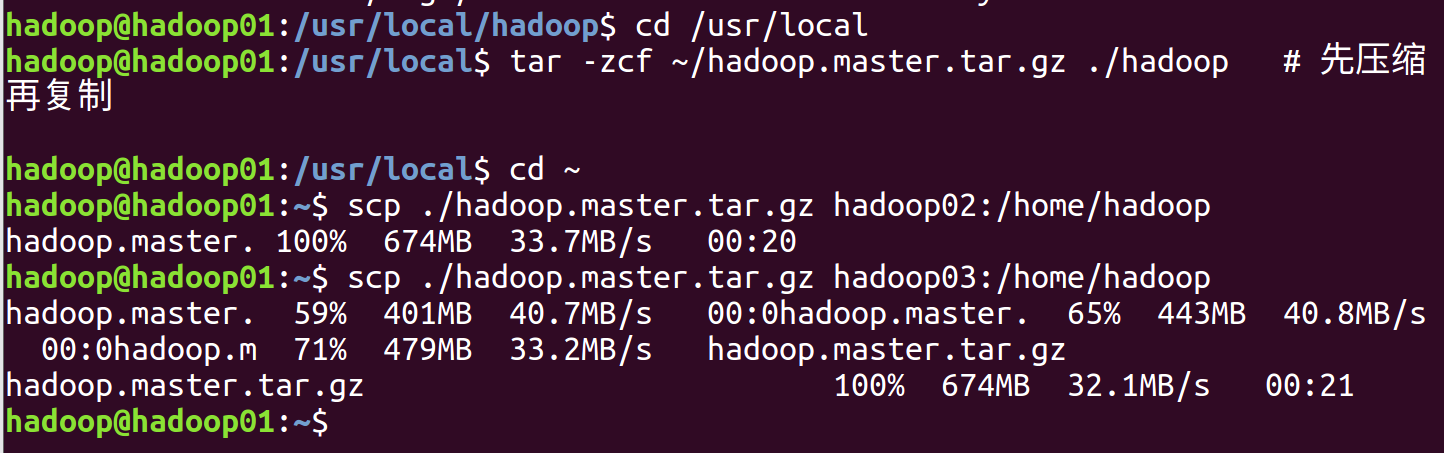


1. 修改这三个节点的“/etc/hosts”文件：



使其互相ping通

1. 重复6.1.3.2的操作，在Master节点（hadoop01）将上公匙传输到Slave节点（hadoop02）和Slave2节点（hadoop03）



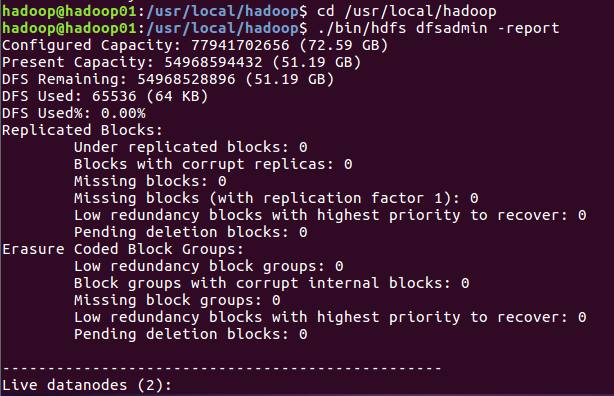
便可以在Master节点上登录到Slave1节点和Slave2节点。

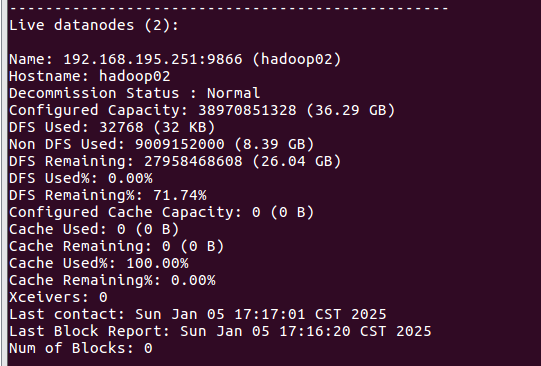
1. 修改/usr/local/hadoop/etc/hadoop/workers文件，修改成：

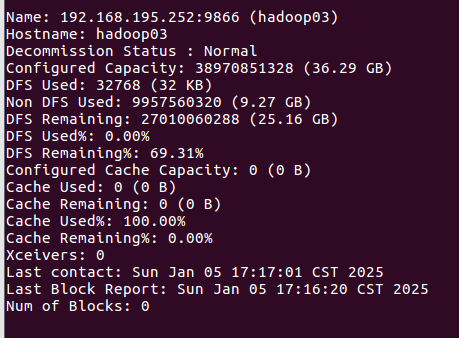
hadoop02

hadoop03

1. 重复6.1.3.5操作，将hadoop.master.tar.gz传输到Slave1节点和Slave2节点并解压文件的操作
2. 在Master节点（hadoop01）执行名称节点的格式化。
3. 启动Hadoop集群（一个主节点两个从节点），结果与6.1.3.5类似，但启动的数据节点数目不同







**6.2数据爬取**

在招聘数据分析中，数据从章鱼公有云平台“对招聘网站分析”的实训板块获取。搜集到的数据包括职位名称、公司名称、薪资范围、工作地点、学历要求、工作经验等。

**6.3数据清洗**

**·BeautifulSoup清洗职位信息网页**

1. **在linux上新建/data/python\_pj2目录。**

|  |
| --- |
| sudo mkdir -p /data/python\_pj2  sudo chown -R hadoop:hadoop /data #将/data目录权限所属调整为hadoop用户及hadoop用户组 |

1. **切换目录到/data/python\_pj2目录下，解压提前下载爬取到的智联招聘数据的jobs.zhaopin.com.tgz。**

|  |
| --- |
| cd /data/python\_pj2  tar -xzvf jobs.zhaopin.com.tgz |

1. **打开Pycharm，若已经创建ZhiLian项目，这里选择“Open”直接打开项目，若没有创建ZhiLian项目，则选择Create New Project，创建名为ZhiLian的项目后创建名为QingXi的Python文件。**
2. **打开QingXi.py文件，现在我们编写代码，清洗智联招聘网站原网页数据。导入程序所用的外包**

|  |
| --- |
| from bs4 import BeautifulSoup  import pandas as pd  import os |

1. **获取网页的有用信息并保存成字典形式**

|  |
| --- |
| def get\_html\_info(html):  '''''  \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*获取网页的有用信息并保存成字典形式\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*  '''  try:  soup = BeautifulSoup(html, "lxml") # 设置解析器为“lxml”  occ\_name = soup.select('div.fixed-inner-box h1')[0]  com\_name = soup.select('div.fixed-inner-box h2 ')[0]  welfare = soup.select('div.welfare-tab-box')[0]  wages = soup.select('div.terminalpage-left strong')[0]  date = soup.select('div.terminalpage-left strong')[2]  exper = soup.select('div.terminalpage-left strong')[4]  num = soup.select('div.terminalpage-left strong')[6]  area = soup.select('div.terminalpage-left strong')[1]  nature = soup.select('div.terminalpage-left strong')[3]  Edu = soup.select('div.terminalpage-left strong')[5]  cate = soup.select('div.terminalpage-left strong')[7]  com\_scale = soup.select('ul.terminal-ul.clearfix li strong')[8]  com\_nature = soup.select('ul.terminal-ul.clearfix li strong')[9]  com\_cate = soup.select('ul.terminal-ul.clearfix li strong')[10]  com\_url = soup.select('div.fixed-inner-box h2 a')[0]  com\_address = soup.select('ul.terminal-ul.clearfix li strong')[-1]  job\_descritions1 = soup.select('div.tab-inner-cont')[0]  job\_descritions=job\_descritions1.select('p')[:-1]  data = {  "工作名称": occ\_name.text.strip(),  "公司名称": com\_name.text,  "公司网址": com\_url.get('href'),  "福利": welfare.text.strip(),  "月工资": wages.text.strip(),  "发布日期": date.text.strip(),  "经验": exper.text.strip(),  "人数": num.text.strip(),  "工作地点": area.text.strip(),  "工作性质": nature.text.strip(),  "最低学历": Edu.text.strip(),  "职位类别": cate.text.strip(),  "公司规模": com\_scale.text.strip(),  "公司性质": com\_nature.text.strip(),  "公司行业": com\_cate.text.strip(),  "公司地址": com\_address.text.strip(),  "岗位描述": [job\_descrition.text.strip() for job\_descrition in job\_descritions],  }  # print(data)  return (data)  except Exception:  pass |

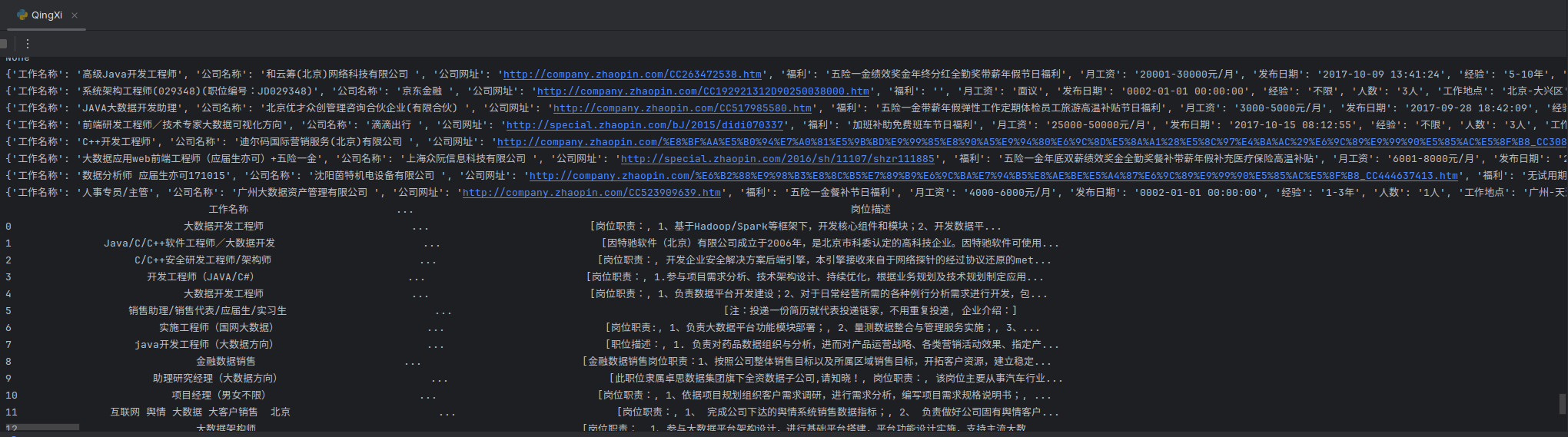
1. **将全部信息保存成矩阵形式，去除无用信息。**

|  |
| --- |
| def get\_htmls\_all\_info(path):  '''''  将全部信息保存成矩阵形式，去除无用信息，并在当前目录下生成文件夹并此文件夹下把信息分类保存成.csv格式  '''  df = pd.DataFrame({  "工作名称": [],  "公司名称": [],  "公司网址": [],  "福利": [],  "月工资": [],  "发布日期": [],  "经验": [],  "人数": [],  "工作地点": [],  "工作性质": [],  "最低学历": [],  "职位类别": [],  "公司规模": [],  "公司性质": [],  "公司行业": [],  "公司地址": [],  "岗位描述": [],  })  dirs = os.listdir(path)  for dir in dirs:  #print(dir)  if dir.find('swp'):  dir=dir.strip('.'and '.swp')  p = os.path.join(path, dir)  html = open(p).read()  data = get\_html\_info(html)  print(data)  df = df.append(data, ignore\_index=True)  return df |

1. **在主函数中调用get\_htmls\_all\_info(path)函数，并把信息保存成.csv格式。**

|  |
| --- |
| if \_\_name\_\_ == '\_\_main\_\_':  path='/data/zhilian/jobs.zhaopin.com'  df=get\_htmls\_all\_info(path)  df.to\_csv('/data/zhilian/bigdata', index=False)  print(df) |

1. **运行QingXi.py**





**6.4数据可视化**

**6.4.1使用PySpark对职位数据进行分析**

1. **开启终端模拟器，创建/data/python\_pj目录**

|  |
| --- |
| sudo mkdir -p /data/python\_pj  sudo chown -R hadoop:hadoop /data/python\_pj # 将/data/python\_pj目录权限所属调整为hadoop用户及hadoop用户组 |

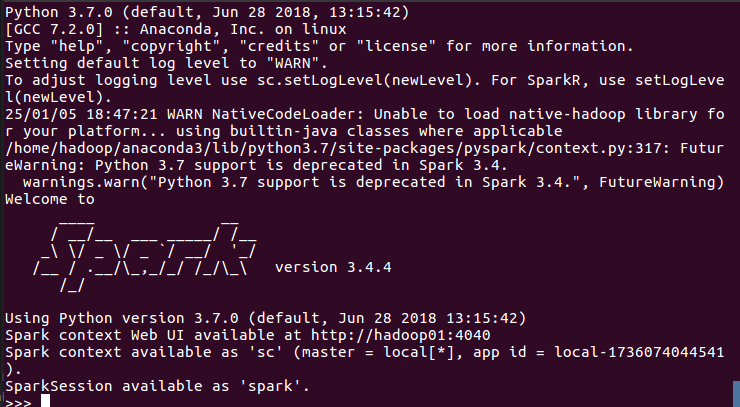
1. **先前由QingXi.py得到的数据bigdata已存放在/data/python\_pj目录下**
2. **启动mysql服务，密码123456**

|  |
| --- |
| sudo service mysql start |

1. **将/data/python\_pj中的bigdata上传到HDFS中的根目录下**

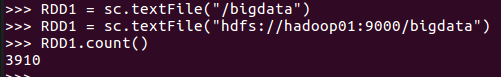
|  |
| --- |
| hdfs dfs -put /data/python\_pj/bigdata / |

1. **开启PySpark**



1. **创建RDD，使用textFile()方法读取HDFS上的bigdata文件，赋值给RDD1并统计文件内容有多少行**

|  |
| --- |
| RDD1 = sc.textFile("hdfs://hadoop01:9000/bigdata")  RDD1.count() |



1. **使用map函数处理每一项数据，用lambda语句创建匿名函数传入line参数，在匿名函数中，line.split(“,”)表示按照逗号分隔获取每一个字段**

|  |
| --- |
| RDD2=RDD1.map(lambda line:line.split(",")) |

1. **创建DataFrame，导入row模块，通过RDD2创建DataFrame，定义DataFrame的每一个字段名与数据类型**

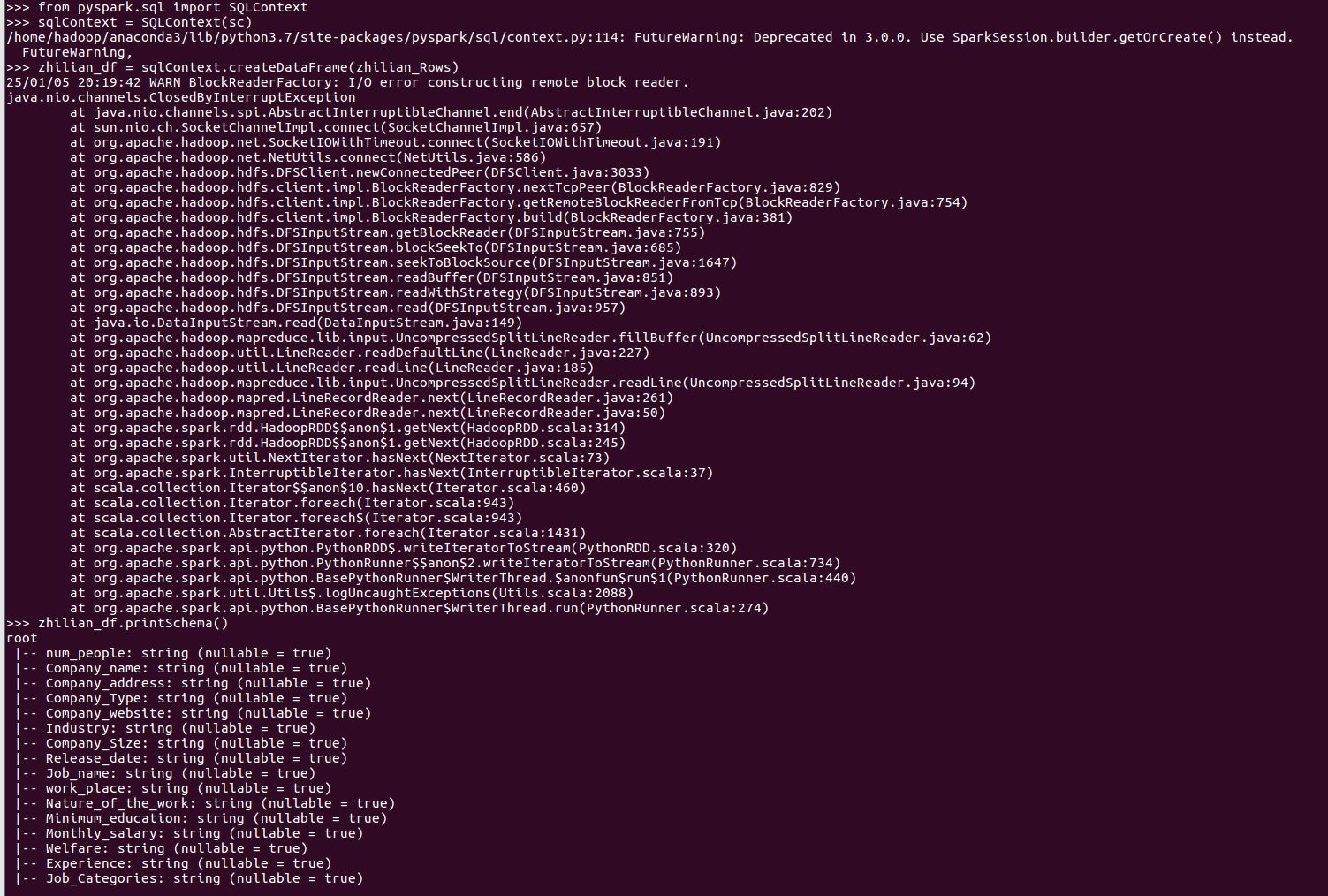
|  |
| --- |
| from pyspark.sql import Row  zhilian\_Rows = RDD2.map(lambda p:  Row(  num\_people=p[0],  Company\_name=p[1],  Company\_address=p[2],  Company\_Type=p[3],  Company\_website=p[4],  Industry=p[5],  Company\_Size=p[6],  Release\_date=p[7],  Job\_name=p[8],  work\_place=p[9],  Nature\_of\_the\_work=p[10],  Minimum\_education=p[11],  Monthly\_salary=p[12],  Welfare=p[13],  Experience=p[14],  Job\_Categories=p[15]  )  ) |

1. **创建了zhilian\_Rows之后，使用sqlContext.createDataFrame()方法写入zhilian\_Rows数据，创建DataFrame，然后使用.printSchema()方法查看DataFrames的Schema**

|  |
| --- |
| from pyspark.sql import Row  zhilian\_Rows = RDD2.map(lambda p:  Row(  num\_people=p[0],  Company\_name=p[1],  Company\_address=p[2],  Company\_Type=p[3],  Company\_website=p[4],  Industry=p[5],  Company\_Size=p[6],  Release\_date=p[7],  Job\_name=p[8],  work\_place=p[9],  Nature\_of\_the\_work=p[10],  Minimum\_education=p[11],  Monthly\_salary=p[12],  Welfare=p[13],  Experience=p[14],  Job\_Categories=p[15]  )  ) |

1. **创建了zhilian\_Rows之后，使用sqlContext.createDataFrame()方法写入zhilian\_Rows数据，创建DataFrame，然后使用.printSchema()方法查看DataFrames的Schema**

|  |
| --- |
| from pyspark.sql import SQLContext  sqlContext = SQLContext(sc)  zhilian\_df = sqlContext.createDataFrame(zhilian\_Rows)  zhilian\_df.printSchema() |

****

1. **接下来，我们可以使用.show()方法来查看前5行数据**

|  |
| --- |
| zhilian\_df.show(5) |



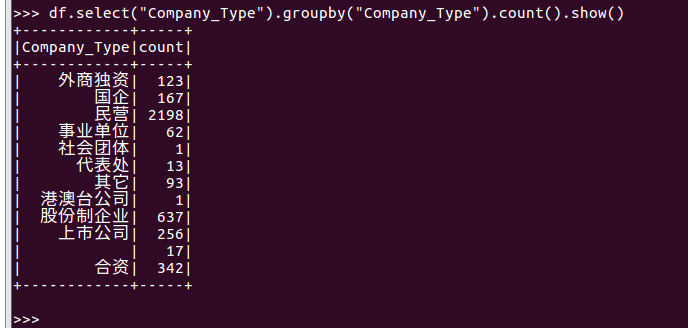
1. **我们也可以使用.alias()方法来为DadaFrame创建别名，例如zhilian\_df.alias("df")，后续我们就可以使用这个别名执行命令了**

|  |
| --- |
| df=zhilian\_df.alias("df")  df.show(5) |



1. **使用DataFrame统计公司性质及数量**

|  |
| --- |
| df.select("Company\_Type").groupby("Company\_Type").count().show() |

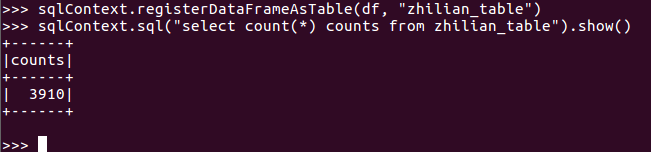


1. **创建PySpark SQL，我们之前创建DataFrame，下面我们使用registerTempTable方法将df转换为zhilian\_table表**

|  |
| --- |
| sqlContext.registerDataFrameAsTable(df, "zhilian\_table") |

1. **接下来，我们可以使用sqlContext.sql()输入sql语句，使用select关键字查询文件内容行数，并使用from关键字指定要查询的表，最后使用show()方法显示查询结果**

|  |
| --- |
| sqlContext.sql("select count(\*) counts from zhilian\_table").show() |

****

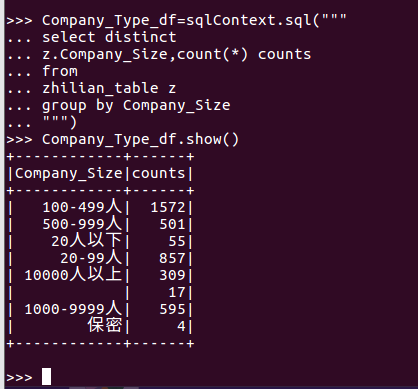
1. **使用PySpark SQL统计经验要求及数量**

|  |
| --- |
| Company\_Type\_df=sqlContext.sql("""  select distinct  z.Experience,count(\*) counts  from  zhilian\_table z  group by Experience  """)  Company\_Type\_df.show() |



1. **使用PySpark SQL统计公司规模及数量**

|  |
| --- |
| Company\_Type\_df=sqlContext.sql("""  select distinct  z.Company\_Size,count(\*) counts  from  zhilian\_table z  group by Company\_Size  """)  Company\_Type\_df.show() |



**6.4.2 对招聘职位信息进行探索分析**

**6.4.2.1 利用matplotlib包里的pyplot作图来分析招聘数据各属性分布情况**

1. **创建explore\_analysis.py，导入各种库**

|  |
| --- |
| import pandas as pd  import matplotlib.pyplot as plt  import numpy as np  import matplotlib.font\_manager as fm  fontPath ="/usr/share/fonts/truetype/wqy/wqy-zenhei.ttc"  font = fm.FontProperties(fname=fontPath, size=10)  data=pd.read\_csv('/data/python\_pj3/bigdata',)  print(data.shape,data.columns)  data.loc[(data.经验=='3年以上'),'经验'] = '3-5年' |

1. **公司规模分布情况**

|  |
| --- |
| plt.figure(figsize=(20,15))  plt.subplot2grid((2,3),(0,0))  a=data['公司规模'].value\_counts().plot(kind='barh',title='公司规模分布情况',color='pink')  a.xaxis.get\_label().set\_fontproperties(font)  a.yaxis.get\_label().set\_fontproperties(font)  a.legend(loc='best',prop=font)  for label in ([a.title]+a.get\_xticklabels()+a.get\_yticklabels()):  label.set\_fontproperties(font)  #公司性质分布情况  plt.subplot2grid((2,3),(0,1))  b=data['公司性质'].value\_counts().plot(kind='barh',title='公司性质分布情况',color='orange')  b.xaxis.get\_label().set\_fontproperties(font)  b.yaxis.get\_label().set\_fontproperties(font)  b.legend(loc='best',prop=font)  for label in ([b.title]+b.get\_xticklabels()+b.get\_yticklabels()):  label.set\_fontproperties(font) |

1. **公司性质分布情况**

|  |
| --- |
| plt.subplot2grid((2,3),(0,1))  b=data['公司性质'].value\_counts().plot(kind='barh',title='公司性质分布情况',color='orange')  b.xaxis.get\_label().set\_fontproperties(font)  b.yaxis.get\_label().set\_fontproperties(font)  b.legend(loc='best',prop=font)  for label in ([b.title]+b.get\_xticklabels()+b.get\_yticklabels()):  label.set\_fontproperties(font) |

1. **经验分布情况**

|  |
| --- |
| plt.subplot2grid((2,2),(1,0),colspan=2)  plt.subplot2grid((2,3),(0,2))  c=data['经验'].value\_counts().plot(kind='barh',title='经验分布情况',color='tomato')  c.xaxis.get\_label().set\_fontproperties(font)  c.yaxis.get\_label().set\_fontproperties(font)  c.legend(loc='best',prop=font)  for label in ([c.title]+c.get\_xticklabels()+c.get\_yticklabels()):  label.set\_fontproperties(font) |

1. **公司行业分布情况**

|  |
| --- |
| plt.subplot2grid((2,3),(1,0))  d=data['公司行业'].value\_counts().sort\_values(ascending=False).head(10).plot(kind='barh',title='公司行业分布情况',color='turquoise')  d.yaxis.get\_label().set\_fontproperties(font)  d.legend(loc='best',prop=font)  for label in ([d.title]+d.get\_xticklabels()+d.get\_yticklabels()):  label.set\_fontproperties(font) |

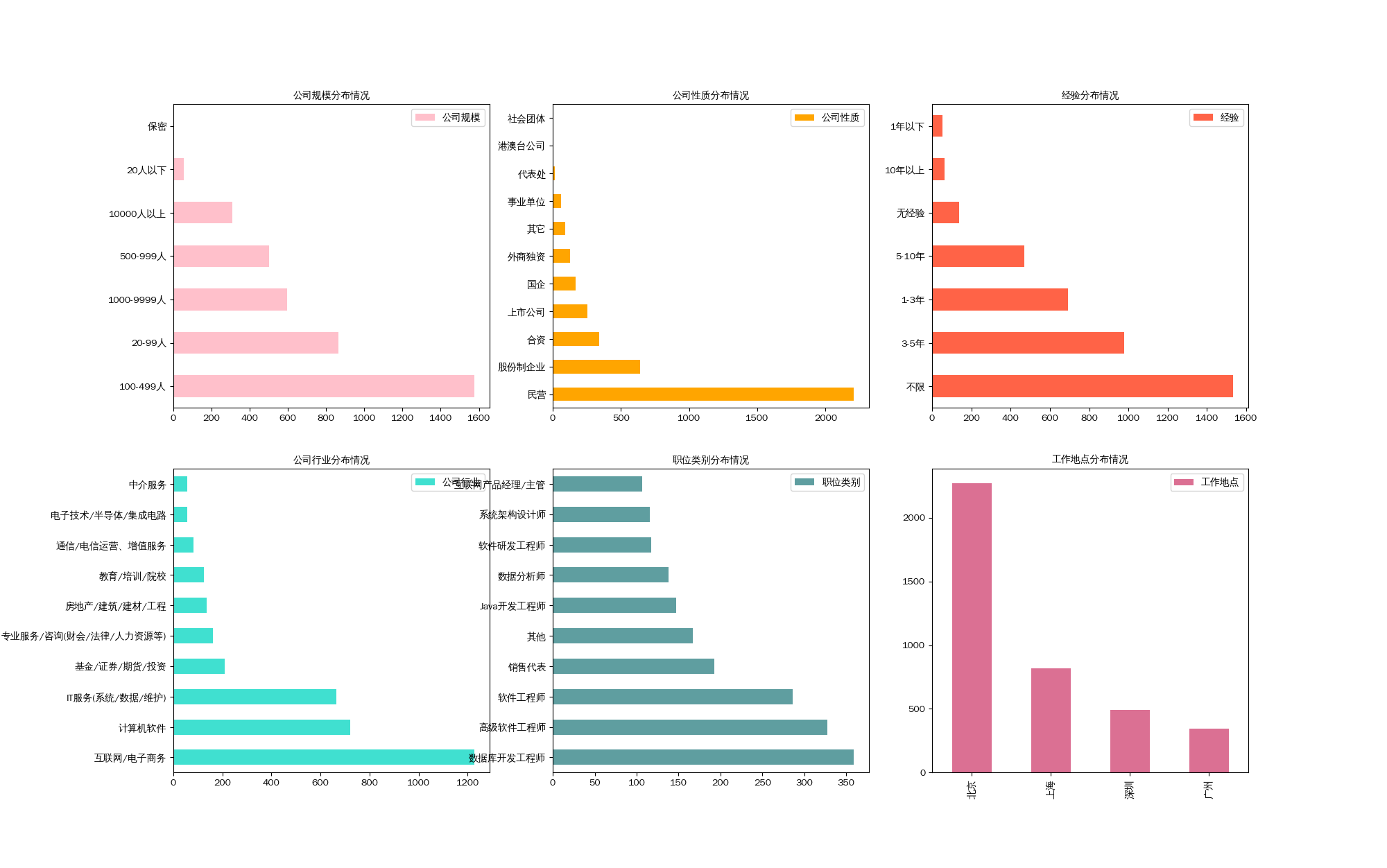
1. **职位类别分布情况**

|  |
| --- |
| plt.subplot2grid((2,3),(1,1))  d=data['职位类别'].value\_counts().sort\_values(ascending=False).head(10).plot(kind='barh',title='职位类别分布情况',color='cadetblue')  d.xaxis.get\_label().set\_fontproperties(font)  d.yaxis.get\_label().set\_fontproperties(font)  d.legend(loc='best',prop=font)  for label in ([d.title]+d.get\_xticklabels()+d.get\_yticklabels()):  label.set\_fontproperties(font) |

1. **工作地点分布情况**

|  |
| --- |
| plt.subplot2grid((2,3),(1,2))  d=data['工作地点'].str.split('-',expand=True)[0].value\_counts().plot(kind='bar',title='工作地点分布情况',color='palevioletred',label='工作地点')  d.xaxis.get\_label().set\_fontproperties(font)  d.yaxis.get\_label().set\_fontproperties(font)  d.legend(loc='best',prop=font)  # print(d.get\_legend\_handles\_labels())  for label in ([d.title]+d.get\_xticklabels()+d.get\_yticklabels()):  label.set\_fontproperties(font)  plt.show() |

1. **explore\_analysis.py展示结果**



由上图可知：

* 公司规模在100-499人的公司招聘的大数据岗位最多。
* 公司性质为民营企业招聘的大数据岗位最多
* 经验要求大部分没有明确说明，剩下的基本上集中在1-5年之间
* 招聘公司主营行业主要集中在互联网、计算机、IT服务等行业
* 职位类别主要侧重于数据库开发，软件工程师等岗位
* 工作地点主要分布在北京，广东最少

**6.4.2.2 利用matplotlib包里的pyplot作图来进行各属性与工资的关联统计**

1. **创建explore\_analysis1.py，导入各种库，处理数据**

|  |
| --- |
| import pandas as pd  import seaborn as sns  import pandas as pd  import seaborn as sns  import matplotlib.pyplot as plt  import numpy as np  import matplotlib.font\_manager as fm  fontPath ="/usr/share/fonts/truetype/wqy/wqy-zenhei.ttc"  font = fm.FontProperties(fname=fontPath, size=10)  data=pd.read\_csv('/data/python\_pj3/bigdata')  print(data.shape)  print(data.columns)  # print([data.职位类别.value\_counts().index if data.职位类别.value\_counts()<10==False:])  data.loc[((data.职位类别=='客户代表')|(data.职位类别=='电话销售')|(data.职位类别=='大客户销售代表')),'职位类别']='销售代表'  月工资=data['月工资'].str.strip('元/月').str.split('-',expand=True)  月工资.columns=['月工资\_min','月工资\_max']  data['月工资\_min']=月工资['月工资\_min']  data['月工资\_max']=月工资['月工资\_max']  data.loc[(data.月工资\_min=='面议'),'月工资\_min']=0  data.loc[(data.月工资\_min=='1000元/月以下' ),'月工资\_min']=1  data.loc[(data.月工资\_min=='100000元/月以上'),'月工资\_min']=7  data.loc[(data.月工资\_max.isnull()),'月工资\_max']=0  print('\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*')  data.月工资\_min=data.月工资\_min.astype(int)  data.月工资\_max=data.月工资\_max.astype(int)  月工资\_mean=(data.月工资\_min+data.月工资\_max)/2  data.loc[((data.月工资\_min==0) & (data.月工资\_max==0) & (data.月工资.notnull()) ),'月工资']=0  data.loc[((data.月工资\_min==1) & (data.月工资\_max==0) & (data.月工资.notnull()) ),'月工资']=1  data.loc[((data.月工资\_min<6000) & (data.月工资\_min>8)),'月工资'] = 1  data.loc[((((data.月工资\_min>=6000) & (data.月工资\_max<=8000)) | ((6000<月工资\_mean)&(月工资\_mean<8000))) & (data.月工资.notnull())),'月工资'] = 2  data.loc[((((data.月工资\_min>=8000) & (data.月工资\_max<=10000)) | ((8000<=月工资\_mean)&(月工资\_mean<10000))) & (data.月工资.notnull()) ),'月工资']=3  data.loc[((((data.月工资\_min>=10000) & (data.月工资\_max<=20000)) | ((10000<=月工资\_mean)&(月工资\_mean<20000))) & (data.月工资.notnull())),'月工资']=4  # data.loc[((data.月工资\_min>=15000) & (data.月工资\_max<=20000) & (data.月工资.notnull()) ),'月工资']=5  data.loc[((((data.月工资\_min>=20000) & (data.月工资\_max<=30000)) | ((20000<=月工资\_mean)&(月工资\_mean<30000))) & (data.月工资.notnull()) ),'月工资']=5  data.loc[((((data.月工资\_min>=30000) & (data.月工资\_max<=50000)) | ((30000<=月工资\_mean)&(月工资\_mean<50000))) & (data.月工资.notnull())),'月工资']=6  data.loc[(((data.月工资\_min>=50000) | (50000<=月工资\_mean)) & (data.月工资.notnull()) ),'月工资']=7  data.loc[((data.月工资\_min==7) & (data.月工资\_max==0) & (data.月工资.notnull()) ),'月工资']=7  print(data.月工资.value\_counts(sort=False)) |

1. **月工资分布情况**

|  |
| --- |
| fig=plt.figure()  fracs=data.月工资.value\_counts(sort=False)  labels=['面议','6000元/月以下','6000-8000元/月','8000-10000元/月','10000-20000元/月','20000-30000元/月','30000-50000元/月','500000元/月以上 ']  colors = ['palevioletred','turquoise','cadetblue','pink','slategrey','orange','green','red']  explode=[0,0,0,0,0.05,0,0,0]  patchs,l\_text,p\_text=plt.pie(x=fracs,explode=explode, labels=labels,colors=colors, autopct='%3.1f %%',  shadow=True, labeldistance=1.1, startangle=0, pctdistance=0.6)  plt.title('月工资分布情况',fontproperties=font)  for t in (l\_text+p\_text):  t.set\_fontproperties(font) |

1. **各经验等级的月工资情况**

|  |
| --- |
| data.loc[(data.经验=='3年以上'),'经验']='3-5年'  print("\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*")  df=pd.DataFrame([data.经验[data.月工资==i].value\_counts().rename(i) for i in range(8)])  print(df)  df.plot(kind='bar',colors=['yellowgreen','lightskyblue','pink','coral','orange','green','palegreen'])  plt.legend(loc='best',prop=font)  plt.xticks(df.index,labels,rotation=15,fontproperties=font)  plt.title('各经验等级的工资分布情况',fontproperties=font)  plt.xlabel('工资等级',fontproperties=font)  plt.ylabel('公司数量',fontproperties=font) |

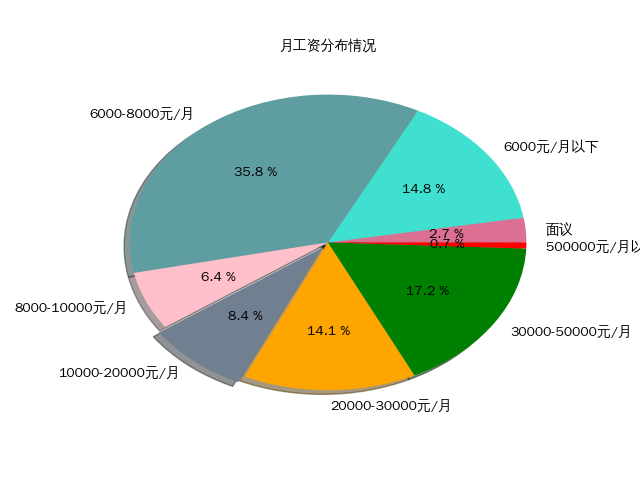
1. **月工资与学历**

|  |
| --- |
| df=pd.DataFrame([data.最低学历[data.月工资==i].value\_counts().rename(i) for i in range(8)])  print(df)  df.plot(kind='bar',stacked=True,colors=colors)  plt.legend(loc='best',prop=font)  plt.xticks(df.index,labels,rotation=15,fontproperties=font)  plt.title('各学历等级的工资分布情况',fontproperties=font)  plt.xlabel('工资等级',fontproperties=font)  plt.ylabel('公司数量',fontproperties=font) |

1. **月工资与工作地点**

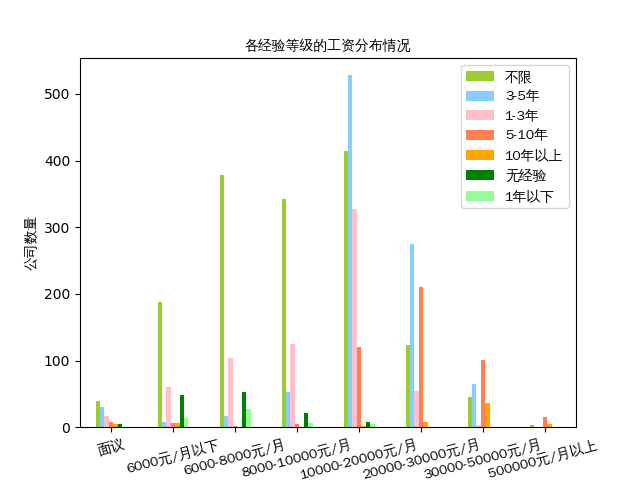
|  |
| --- |
| df=pd.DataFrame([data.工作地点.str.split('-',expand=True)[0][data.月工资==i].value\_counts().rename(i) for i in range(8)]).T  print(df)  colors=['burlywood','yellowgreen','lightskyblue','pink','coral','orange','plum','oldlace']  d=df.plot(kind='bar',stacked=True,colors=colors)  plt.legend(loc='best',prop=font,labels=labels)  print(d.get\_legend\_handles\_labels())  plt.xticks([i for i in range(len(df.index))],df.index,rotation=15,fontproperties=font)  plt.title('工作地点的工资分布情况',fontproperties=font)  plt.xlabel('工资等级',fontproperties=font)  plt.ylabel('公司数量',fontproperties=font)  print(df.values)  plt.show() |

1. **explore\_analysis1.py运行结果**
   1. **月工资分布情况**



由上图可知：公司招聘时，给的月工资主要集中在10000-20000元/月为35.8%，其次是20000-30000，10000-8000或6000-8000的也占很大比重，50000元/月以上的所占比例最小为0.7%，几乎可以省略。

* 1. **各经验等级的工资分布情况**



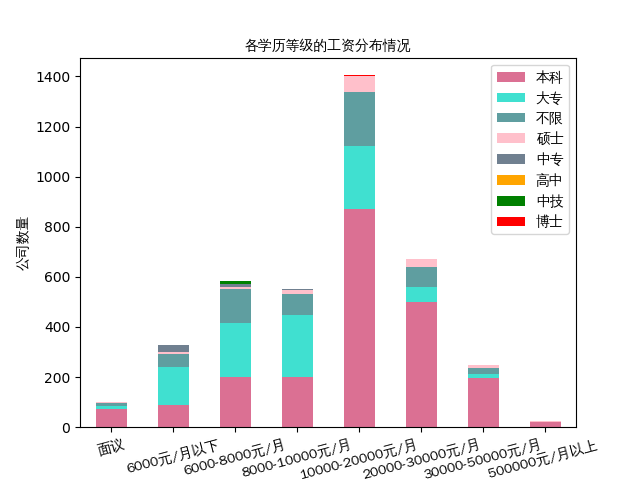
由于部分公司在智联发布消息时，会把经验要求写到职位要求里，经验框中写不限经验，所以我们对经验框中填写不限经验的不做分析。

由上图可知：

* 薪资在1万元/月以下，经验要求由高到低的是1-3年，1-3年、无经验
* 薪资在1-2万元/月，经验要求由高到低的是3-5年，1-3年、5-10年
* 薪资在2-3万元/月，经验要求由高到低的是3-5年，5-10年、1-3年
* 薪资在3-5万元/月，经验要求由高到低的是5-10年，3-5年、10年以上
* 薪资在5万元/月以上，经验要求由高到低的是5-10年，3-5年、10年以上。

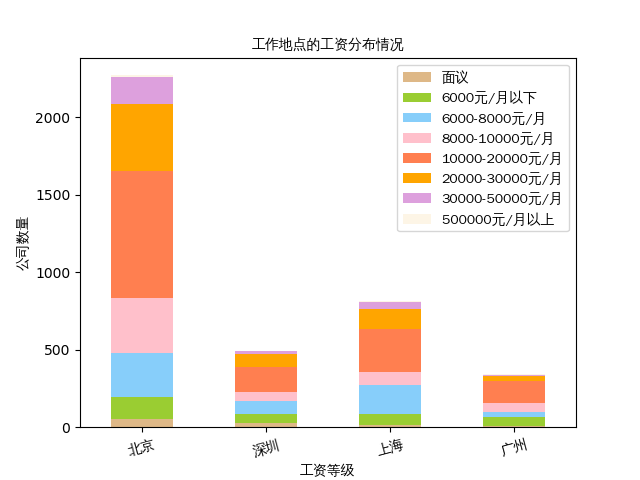
由此可得，随着经验的增加，薪资在逐步的提高，当经验达到3年时，基本上能达到1万以上。

* 1. **各学历等级的工资分布情况**



由上图可知：在各薪资范围内要求最低学历为本科的公司最多，中专、中技、高中几乎不存在。

* 1. **工作地点的工资分布情况**



由图可知：各城市公司招聘给定的薪资10000-20000元/月所占比例最大，不同薪资分布比例基本较相似。

**6.4.3使用结巴分词对岗位描述进行分词并将关键词**

1. **创建JieBa\_FenCi.py文件，现在我们编写代码，使用结巴分词对岗位描述中的提及到的关键技术进行词频统计。导入代码所需的外包**

|  |
| --- |
| import pandas as pd  import re  import numpy as np  import jieba  import jieba.analyse  import pymysql  from sqlalchemy import create\_engine |

1. **使用pandas中的方法,过滤掉岗位描述列中不符合的内容**

|  |
| --- |
| job\_des=data.岗位描述.str.strip("[]").str.replace(',','').\  str.replace("'","").str.replace(r'\\xa0','').\  str.replace(' '\*3 or ' '\*4,'|').str.split('|',expand=True)[0]  # print(job\_des) |

1. **jieba分词，并统计词频，将结果保存到字典中**

|  |
| --- |
| zidian={}  r1=re.compile(r'\w') #使用正则表达式,筛选[A-Za-z0-9\_]  r2=re.compile(r'[^\d]') #使用正则表达式,筛选[0-9\_]  r4=re.compile(r'[^\_]')  r3=re.compile(r'[\u4e00-\u9fa5]') #使用正则表达式,筛选纯中文  stopkeyword=[line.strip() for line in open('/home/python\_pj4/stopword').readlines()] #加载停用词  bogs=[] #用于存储长的英文语句  for i in job\_des:  seg\_list = jieba.cut(i)  # print('Dafault Mode:', ' '.join(seg\_list))  for word in seg\_list:  if word not in stopkeyword and r1.match(word) and r2.match(word)and r3.match(word)==None and r4.match(word): #筛选语句中的英文  word=word.lower() #小写化所有词  if len(word)>=10:  bogs.append(word)  elif len(word)!=1 or word in ['c','r']:  #统计词频  if word in zidian:  zidian[word]+=1  else:  zidian[word]=1 |

1. **用于分割长语句中的关键词**

|  |
| --- |
| def cut(sentences,words):  itr=[]  for st in sentences:  for word in words:  if st.find(word)!=-1:  itr.append(word)  cut(st.split(word),words)  return itr |

1. **将分割词的词频累计到‘zidian’中**

|  |
| --- |
| word\_bogs=cut(bogs,df.skill\_name)  for word\_bog in word\_bogs:  if word\_bog in list(zidian.skill\_name):  zidian.loc[zidian.skill\_name==word\_bog,'num']+=1 |

1. **将zidian中数据导入到mysql数据库中**

|  |
| --- |
| conn=create\_engine('mysql+mysqldb://root:strongs@localhost:3306/zhilian?charset=utf8')  pd.io.sql.to\_sql(zidian,name='job\_des',con=conn,schema='zhilian',if\_exists='append',index=False) |

1. **代码编写完毕，切换到Linux终端，输入以下命令启动MySQL服务**

|  |
| --- |
| sudo service mysql start |

1. **进入MySQL库，密码为：strongs**

|  |
| --- |
| mysql -u root -p |

1. **创建zhilian库**

|  |
| --- |
| create database zhilian default charset utf8 collate utf8\_general\_ci; |



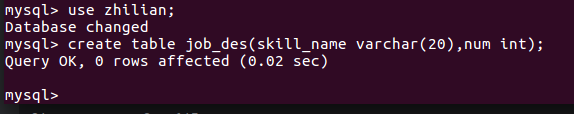
1. **切换到zhilian库**

|  |
| --- |
| use zhilian; |



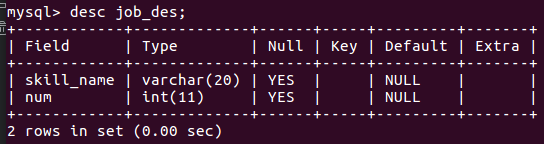
1. **创建表job\_des**

|  |
| --- |
| create table job\_des(skill\_name varchar(20),num int); |

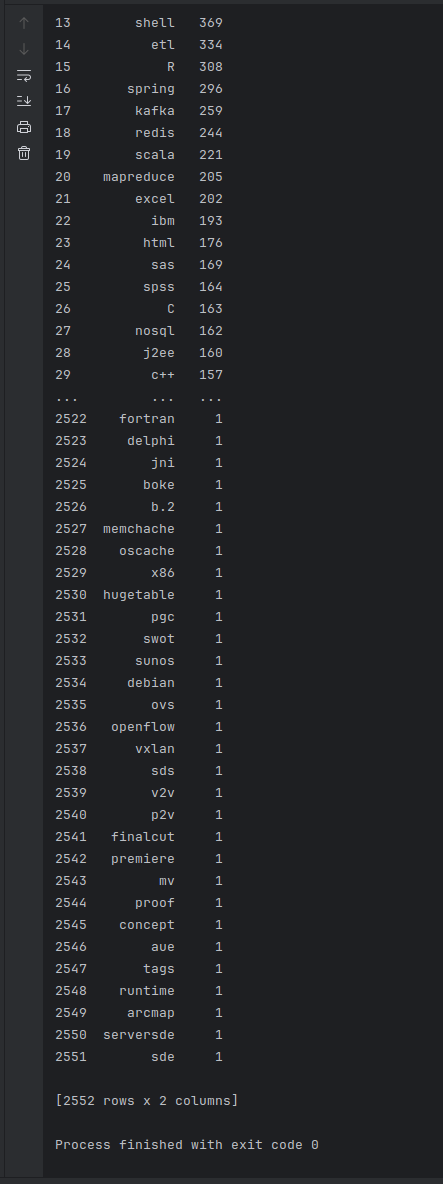


1. **查看job\_des表数据结构**

|  |
| --- |
| desc job\_des; |

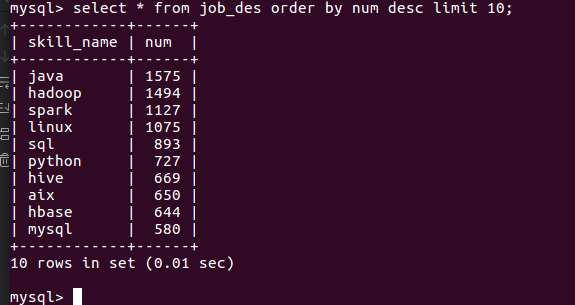


1. **运行JieBa\_FenCi.py文件**



1. **在MySQL中输入下面命令，查看num最大的前十行。要求**

|  |
| --- |
| select \* from job\_des order by num desc limit 10; |

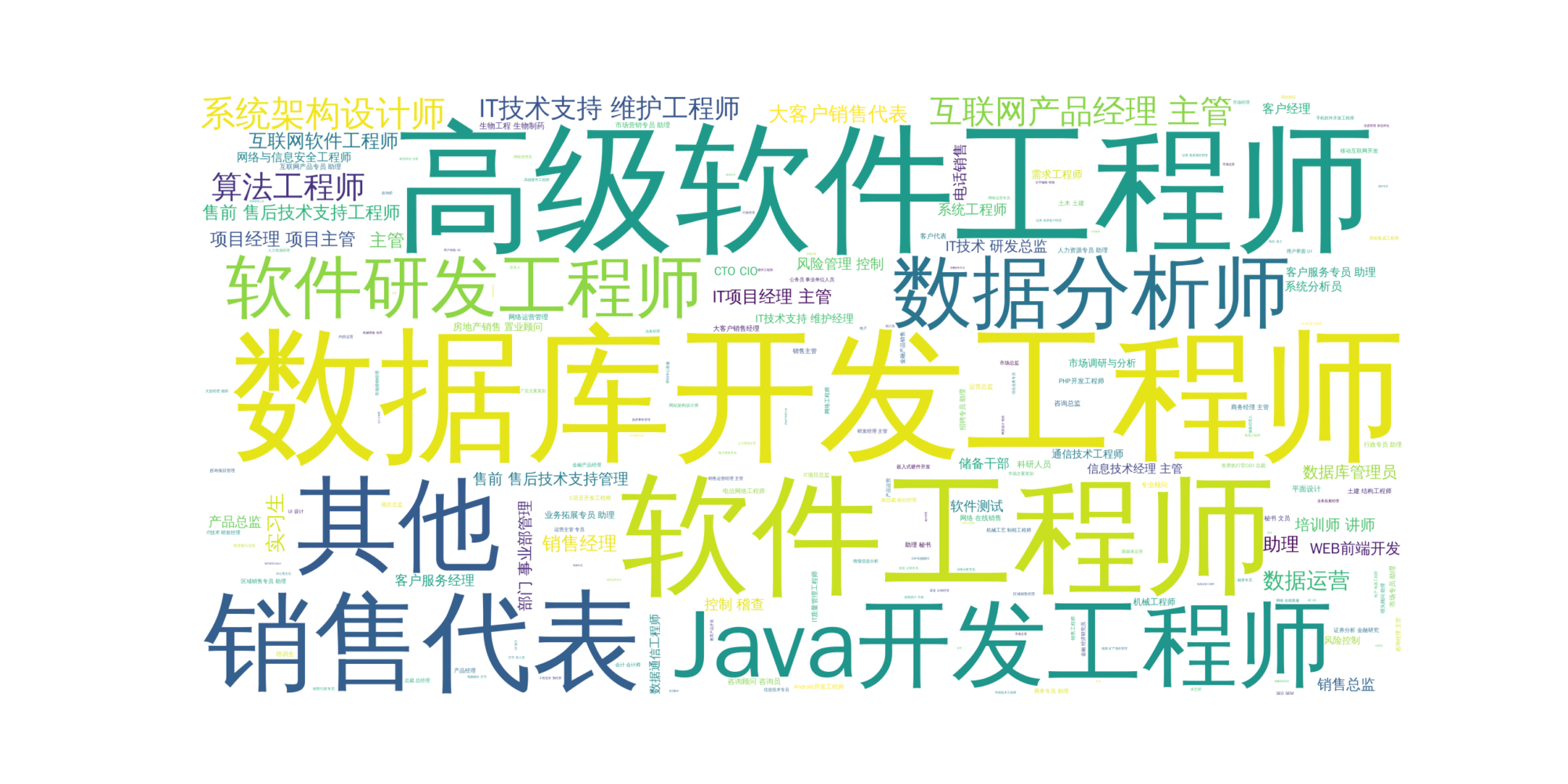


**6.4.4 利用词云得出岗位需求**

1. **创建Job\_Word\_Cloud.py文件，导入各种库**

|  |
| --- |
| import pandas as pd  import matplotlib.pyplot as plt  import numpy as np  import matplotlib.font\_manager as fm  fontPath ="/usr/share/fonts/truetype/wqy/wqy-zenhei.ttc"  font = fm.FontProperties(fname=fontPath,size=10)  data=pd.read\_csv('/data/python\_pj3/bigdata')  from wordcloud import WordCloud  lables = pd.DataFrame(data['职位类别'])  text = ' '.join(lables['职位类别'])  wc = WordCloud(width=2000,height=1000,background\_color='white',font\_path=fontPath).generate(text)  plt.figure(figsize=(20,10)) |

1. **运行Job\_Word\_Cloud.py文件，运行结果展示**



**6.5 数据分析**

1. **创建Random\_Forest.py文件**

|  |
| --- |
| import pandas as pd  import numpy as np  import matplotlib.pyplot as plt  import matplotlib.font\_manager as fm  from sklearn.preprocessing import OneHotEncoder  from sklearn.model\_selection import train\_test\_split  from sklearn.preprocessing import LabelEncoder  from sklearn.ensemble import RandomForestRegressor  fontPath ="/usr/share/fonts/truetype/wqy/wqy-zenhei.ttc"  font = fm.FontProperties(fname=fontPath, size=10)  data=pd.read\_csv('/data/python\_pj3/bigdata')  data = data.dropna()  data = data.reset\_index()  new\_data = data[['公司规模','公司性质','经验','公司行业','职位类别','工作地点','最低学历','月工资']] |

1. **为预测薪资范围，我们目标是将数据清洗数据类型为列表，[XXX,XXX]区间形式的数据，通过查看value.counts( )，我们将月工资为“面议”的数据进行删除，对薪资为“XX以下”的数据整理成[0,XXX]，对于其他数据，利用split( )和apply(lambda x: )方法，将数据整理成[XXX,XXX]的区间形式，方便后续的模型预测。**

|  |
| --- |
| # 自定义一个函数来处理字符串  new\_data = new\_data[new\_data['月工资'] != '面议']  new\_data = new\_data.reset\_index()  def split\_salary\_range(salary):  if salary.endswith('元/月以下'):  # 移除'元/月以下'并转换为整数，然后返回一个列表，包含0和该数字  return [0, int(salary[:-6])]  else:  # 分割范围并转换为整数列表  parts = salary.replace('元/月', '').split('-')  if len(parts) == 2:  start, end = map(int, parts)  return [start, end]  else:  # 如果数据格式不正确  return None  # 应用自定义函数到DataFrame的列  new\_data['月工资'] = new\_data['月工资'].apply(split\_salary\_range)  new\_data = new\_data[new\_data['月工资'].notna()].reset\_index() |

1. **首先初始化一个LabelEncoder对象，用于将分类数据转换为整数。最后初始化一个OneHotEncoder对象，并将之前用LabelEncoder转换过的整数列进一步转换为独热编码。使用np.hstack函数将所有独热编码的特征矩阵水平堆叠起来，形成最终的特征矩阵X，为后续的机器学习模型训练做准备。**

|  |
| --- |
| new\_data = new\_data.astype(str)  # 初始化LabelEncoder  label\_encoder = LabelEncoder()  # 将'最低学历'列的字符串转换为整数  tmp\_data = pd.DataFrame()  tmp\_data['公司规模'] = label\_encoder.fit\_transform(new\_data['公司规模'])  tmp\_data['公司性质'] = label\_encoder.fit\_transform(new\_data['公司性质'])  tmp\_data['经验'] = label\_encoder.fit\_transform(new\_data['经验'])  tmp\_data['公司行业'] = label\_encoder.fit\_transform(new\_data['公司行业'])  tmp\_data['职位类别'] = label\_encoder.fit\_transform(new\_data['职位类别'])  tmp\_data['工作地点'] = label\_encoder.fit\_transform(new\_data['工作地点'])  tmp\_data['最低学历'] = label\_encoder.fit\_transform(new\_data['最低学历'])  # 使用OneHotEncoder  onehot = OneHotEncoder(sparse=False)  company\_size\_onehot = onehot.fit\_transform(tmp\_data[['公司规模']])  company\_property\_onehot = onehot.fit\_transform(tmp\_data[['公司性质']])  experience\_onehot = onehot.fit\_transform(tmp\_data[['经验']])  company\_industry\_onehot = onehot.fit\_transform(tmp\_data[['公司行业']])  job\_category\_onehot = onehot.fit\_transform(tmp\_data[['职位类别']])  work\_place\_onehot = onehot.fit\_transform(tmp\_data[['工作地点']])  education\_onehot = onehot.fit\_transform(tmp\_data[['最低学历']]) |

1. **使用了train\_test\_split函数来将数据集X和标签y分割成训练集和测试集。test\_size=0.01表示测试集占总数据的1%，random\_state=42用于确保每次分割的结果都是一致的。创建了一个随机森林回归器RFM，其中n\_estimators=100表示森林中树的数量为100，random\_state=1用于确保模型的可重复性。**

|  |
| --- |
| salay=np.array([eval(i)for i in new\_data['月工资']])  X = np.hstack((company\_size\_onehot, company\_property\_onehot, experience\_onehot, company\_industry\_onehot,  job\_category\_onehot,work\_place\_onehot,education\_onehot)) # skills\_padded,  y = salay    # 分割数据集  X\_train, X\_test, y\_train, y\_test = train\_test\_split(X, y, test\_size=0.01, random\_state=42)  RFM = RandomForestRegressor(n\_estimators=100, random\_state=1)  RFM.fit(X, salay)  # 进行预测  y\_pred = RFM.predict(X\_test) |

1. **创建一个字典result\_data来存储真实值和预测值，并打印出来。导入mean\_squared\_error函数来计算均方误差（MSE）。计算MSE，并进一步计算根均方误差（RMSE），即MSE的平方根。**
2. **使用plt.scatter绘制真实值和预测值的散点图。使用plt.plot绘制一条完美拟合线，即y=x的线，表示理想情况下的预测值与真实值完全一致。**

|  |
| --- |
| # 计算均方误差  mse = mean\_squared\_error(y\_test, y\_pred)  rmse = np.sqrt(mse)  # 绘制散点图  plt.scatter(y\_test, y\_pred)  plt.plot([y.min(), y.max()], [y.min(), y.max()], 'k--', lw=2) # 绘制完美拟合线  plt.xlabel('Actual Values')  plt.ylabel('Predicted Values')  plt.title('Actual vs Predicted Values')  plt.show() |

1. **结果展示**

