Homework 5: Stock Index Prediction

一、模型建立与数据预处理

线性模型

对于本周的作业，共给了500多个公司的股票数据，用来做线性模型的预测任务。

- 任务：对于每只股票，用线性模型去预估下一天的open价格。
- 数据：共有505个公司，csv格式。

数据预处理

根据`spark MLlib`的数据格式要求，我们需要首先将数据变成如下格式：

```
label,feature1 feature2 ... featureN
```

`label`和`feature`之间用逗号隔开，各个`feature`之间使用空格隔开。

数据集中一共有505家公司的数据，我们对每个公司的数据进行分别处理。代码如下：

```
data_root = './individual_stocks_5yr/'
data_files = os.listdir(data_root)
outdata_root = './out_data/'

for file in data_files:
    ori_file = os.path.join(data_root, file)
    out_file = os.path.join(outdata_root, file)
    df = pd.read_csv(ori_file) # 使用pandas读入csv文件
    for idx in range(len(df)):
        if idx < len(df) - 1:
            df.at[idx, 'label'] = df.at[idx+1, 'open']  # 把后一天的open值当做前一天
的label
    df = df[:-1]  # 忽略掉最后一天的数据
    # 把数据按格式写入新文件
    with open(out_file, 'w') as f:
        for idx, row in df.iterrows():
            f.write(str(row['label'])+',')
            f.write(str(row['open'])+' '+str(row['high'])+' '
                    +str(row['low'])
                    +' '+str(row['close'])+'\n')
```

在这里，我们使用了一个简单可行的想法：把后一天的open价格当做前一天label，然后使用线性回归进行拟合。

特征提取

由于这是一个时序预测的任务，我们将下一天的open价格当作当前的label值。同时将name、date、volume等无关的属性剔除。清洗后的数据如下所示：

```
label, open high low close # label是下一天的open值
```

然后调用MLlib的线性模型包去进行模型的运算。

二、算法实现

1. 代码理解

我们参考了spark官网mllib介绍文档中的代码，代码如下

```
import org.apache.spark.sql.SparkSession
import org.apache.spark.mllib.regression.LinearRegressionWithSGD
import org.apache.spark.mllib.regression.LabeledPoint
import org.apache.spark.mllib.linalg.Vectors

object linear {
  def main(args: Array[String]) {
    val inputPath = args(0)
    val iterations = args(1).toInt

    val spark = SparkSession
      .builder
      .appName("linear")
      .getOrCreate()

    // Load and parse the data
    val data = spark.read.textFile(inputPath).rdd
    val parsedData = data.map { line =>
      val parts = line.split(',')
      LabeledPoint(parts(0).toDouble, Vectors.dense(parts(1).split('
').map(_.toDouble)))
    }.cache()

    // Building the model
    val model = LinearRegressionWithSGD.train(parsedData, iterations)

    // Evaluate model on training examples and compute training error
    val valuesAndPreds = parsedData.map { point =>
      val prediction = model.predict(point.features)
      (point.label, prediction)
    }
    val MSE = valuesAndPreds.map{case(v, p) => math.pow((v - p), 2)}.mean()
    println("training Mean Squared Error = " + MSE)

    spark.stop()
  }
}
```

代码从输入参数中分别提取出输入路径和迭代次数，随后读入数据，并按照文件格式转化为Scala可处理数据，随后利用LinearRegressionWithSGD训练，得到模型后重新预测，并计算均方误差。

2. 代码实现

在上一次作业中，我们为了使用spark，已经安装了Scala，spark和sbt，在此次作业中，我们调用了mllib，因此只需要额外引用即可使用。

修改build.sbt，增加调用spark-mllib

```
name := "linear-model"

version := "0.1"

scalaVersion := "2.11.7"

libraryDependencies += "org.apache.spark" %% "spark-core" % "2.2.0" % "provided"
libraryDependencies += "org.apache.spark" %% "spark-sql" % "2.2.0" % "provided"
libraryDependencies += "org.apache.spark" %% "spark-mllib" % "2.2.0" % "provided
"
~
~
~
~
~
~
~
~
~
~
~
~
~
                                                                1,1           All
```

随后，进行编译和打包

```
cd ~/ML/linear-regression
sbt
compile
package
```

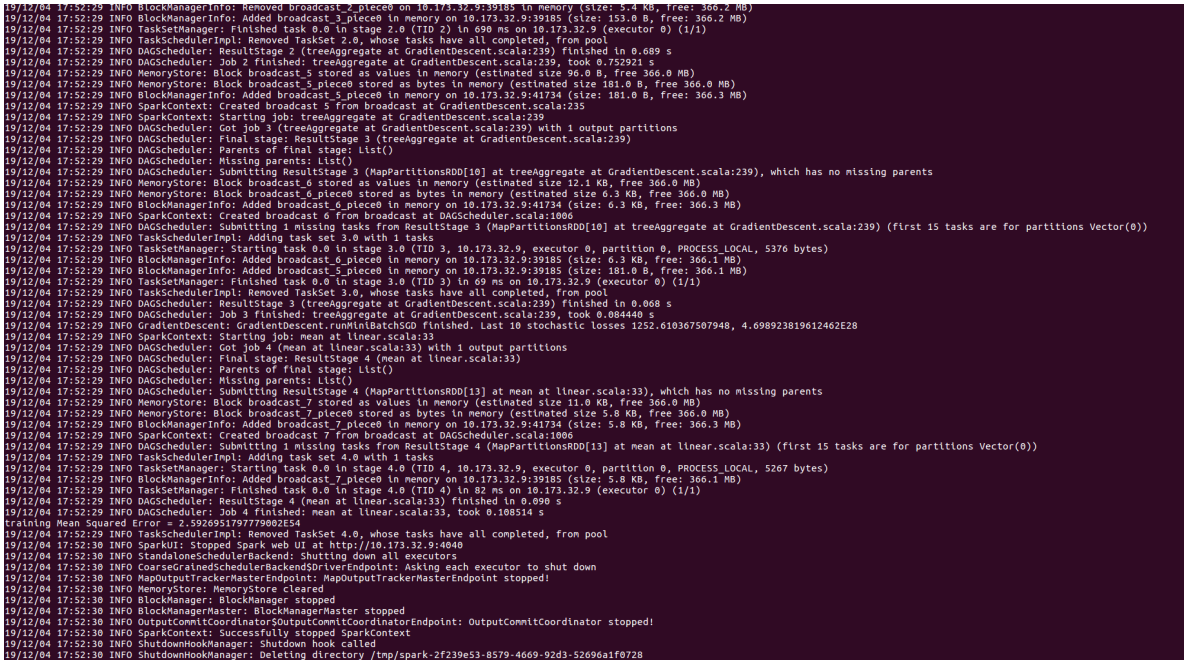
在编译时，sbt会下载相应的包，花费较多的时间。

运行时，目录切换到spark安装目录的bin文件夹下：

```
./spark-submit --master spark://hadoop1:7077 --class linear
/root/LM/linear-model/target/scala-2.11/linear-model_2.11-0.1.jar
~/LM/A_data.txt 2
```

3. 实际运行

我们以hadoop1作为master结点，hadoop1和hadoop2作为slave结点，运行上述代码：

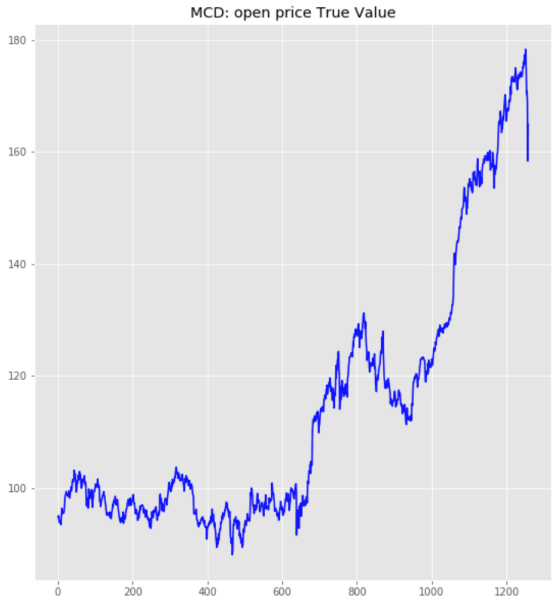


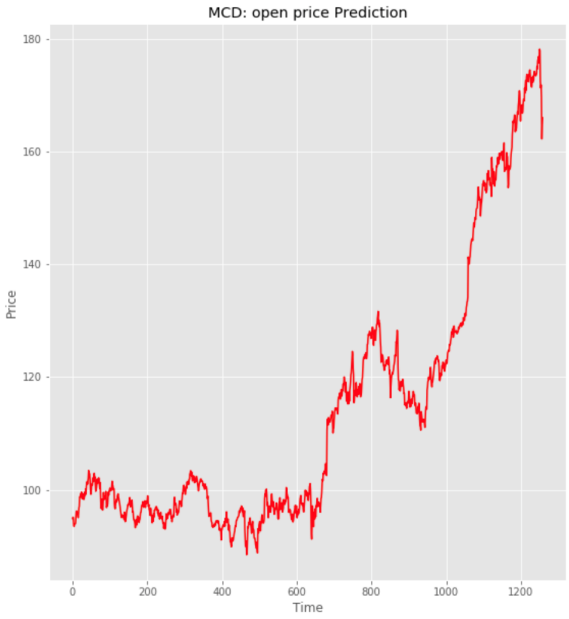
程序成功运行！

三、预测结果

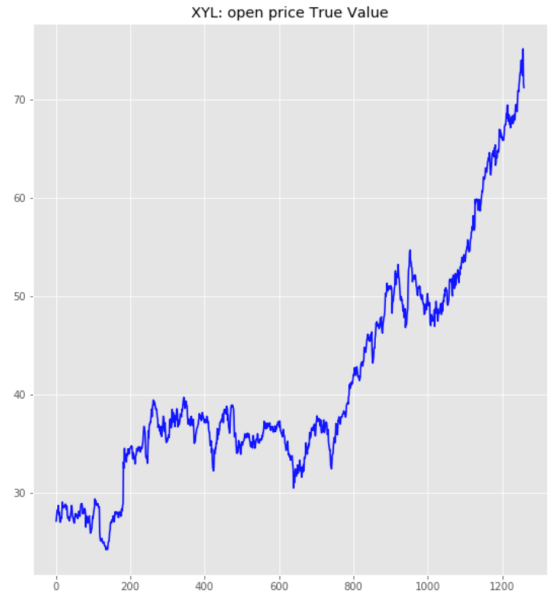
我们将得到的数据导入到python进行可视化处理，以两个公司为例进行预测结果的分析，得到的曲线如下：

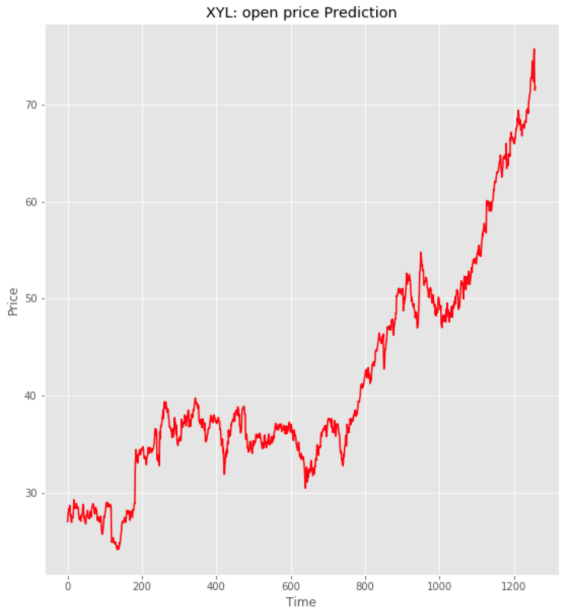
```
MSE=
回归参数为： [[-0.09101293  0.1073159   0.19476493  0.79062517]]
预测值个数： 1259
```






```
MSE=
回归参数为： [[ 4.65455103e-02 -7.24274493e-05 -4.22851507e-02  9.96656652e-01]]
预测值个数： 1259
```





可以看到我们的模型预测的效果较好，可以从两方面来说明：

一、我们的特征选取较为合理，因为众所周知，股票在开市会进行竞拍，竞拍的结果会导致股票价格的起伏，如果没有竞拍，股票的开市价格与前一天的闭市价格相同，因此很大程度上受闭市价格影响，从我们的回归参数也可以看到，最后一项约等于1，其他项的影响较小；

二、我们的测试数据集和训练集重复，我们用了五年的数据进行训练，并将训练得到的模型又用在其中的数据上预测，结果自然不会太差。

四、组员分工

胡晨旭：数据预处理，建立模型

王宇琪：代码处理，数据可视化

张智为：算法运行