大数据实习总结

一. 目录

二. 实习内容分析

1. 实习公司介绍



东软面向全球市场提供IT驱动的创新型解决方案与服务，致力于推动社会的发展与变革，为个人创造新的生活方式，为社会创造价值。公司创立于1991年，目前拥有近20000名员工，在中国建立了8个区域总部，10个软件研发基地， 16个软件开发与技术支持中心，在60多个城市建立营销与服务网络; 在美国、日本、欧洲、中东、南美设有子公司。东软是中国第一个上市的软件公司，也是最先通过CMM5和CMMI(V1.2)5级认证的中国软件公司。

东软提供行业解决方案和产品工程解决方案以及相关软件产品、平台及服务。行业解决方案涵盖领域包括：电信、能源、金融、政府、制造业、商贸流通业、医疗卫生、教育与文化、交通、移动互联网、传媒、环保等。在汽车电子、智能终端、数字家庭产品、IT产品等产品工程领域，东软嵌入式软件服务于众多全球知名品牌产品, 同时，拥有自有品牌的医疗和网络安全产品。在服务领域，东软提供包括应用开发和维护、ERP实施与咨询服务、专业测试及性能工程服务、软件全球化与本地化服务、IT基础设施服务、业务流程外包（BPO）、IT教育培训等。

东软将“超越技术”的经营思想和品牌承诺，关注人与社会发展，致力于成为受社会、客户、股东和员工尊敬的公司。

2. Scala学习总结

Scala 语言和其它语言比如 Java 相比，算是一个比较复杂的语言，它是一个面向对象和面向函数的统一体，使用起来非常灵活，因此也不容易掌握。

Scala可以通过交互式的Scala解释器，也可以通过集成开发环境eclipse进行编写。这2种环境我都用过，前期的一些简单代码就在交互式环境下运行，后来随着代码复杂性的提高，我开始转向eclipse。

Scala的基础语法和Java类似，Scala是大小写敏感的，这意味着标识Hello 和 hello在Scala中会有不同的含义。对于所有的类名，第一个字母要大写。如果需要使用几个单词来构成一个类的名称，每个单词的第一个字母要大写。所有的方法名称，第一个字母要小写。如果若干单词被用于构成方法的名称，则每个单词的第一个字母应大写。程序文件的名称应该与对象名称完全匹配。保存文件时，应该保存它使用的对象名称（Scala是区分大小写），并追加“.scala”为文件扩展名。 （如果文件名和对象名称不匹配，程序将无法编译）。Scala程序从main()方法开始处理，这是每一个Scala程序的强制程序入口部分。Scala 可以使用两种形式的标志符，字符数字和符号。Scala支持单行和多行注释非常类似于Java。Scala 使用 import 关键字引用包，如果想要引入包中的几个成员，可以使用selector（选取器）。

Scala中控制语句和Java差不多，但是循环语句要复杂很多。在Java或C++，把表达式和语句看做两样不同的东西，表达式有值，语句执行动作。但在Scala中，几乎所有构造出来的语法结构都有值，这个特性使得程序更加精简，也更易读。Scala提供了几种循环类型：while循环，do……while循环，for循环。Scala中的for循环功能及其强大，在for循环中可以使用分号 (;) 来设置多个区间，它将迭代给定区间所有的可能值；for循环也可以遍历一个集；在for循环中也可以设置过滤。如果for循环的循环体以yield开始，则该循环会构造出一个集合，每次迭代生成集合中的一个值，这类循环叫for推导式。

Scala字符串和数组类似于Java，在 Scala 中，字符串的类型实际上是 Java String，它本身没有 String 类。String 是一个不可变的对象，所以该对象不可被修改。这就意味着如果修改字符串就会产生一个新的字符串对象。如果你需要创建一个可以修改的字符串，可以使用 String Builder 类。数组是用来存储固定大小的同类型元素，数组中某个指定的元素是通过索引来访问的，数组的第一个元素索引为0，最后一个元素的索引为元素总数减1。

Scala是面向函数的编程，函数是一组一起执行一个任务的语句,可以把代码划分到不同的函数中。在逻辑上，划分通常是根据每个函数执行一个特定的任务来进行的。方法定义由一个def 关键字开始，紧接着是可选的参数列表，一个冒号"：" 和方法的返回类型，一个等于号"="，最后是方法的主体。return type 可以是任意合法的 Scala 数据类型，参数列表中的参数可以使用逗号分隔。在Scala中，可以在变量中存放函数，如果函数没有返回值，可以返回为 Unit，这个类似于 Java 的void。一般情况下函数调用参数，就按照函数定义时的参数顺序一个个传递。但是我们也可以通过指定函数参数名，并且不需要按照顺序向函数传递参数。Scala 允许指明函数的最后一个参数是可变的，即不需要指定函数参数的个数，可以向函数传入可变长度参数列表。Scala 通过在参数的类型之后放一个星号来设置可变参数。Scala 可以为函数参数指定默认参数值，使用了默认参数，在调用函数的过程中可以不需要传递参数，这时函数就会调用它的默认参数值，如果传递了参数，则传递值会取代默认值。可以在 Scala 函数内定义函数，定义在函数内的函数称之为局部函数。Scala的解释器在解析函数参数时有两种方式：传值调用和传名调用。 Scala 中定义匿名函数，箭头左边是参数列表，右边是函数体。高阶函数（Higher-Order Function）就是操作其他函数的函数。当将一个匿名函数传递给另一个函数时，Scala会尽可能推断出类型信息。闭包是一个函数，返回值依赖于声明在函数外部的一个或多个变量。柯里化(Currying)指的是将原来接受两个参数的函数变成新的接受一个参数的函数的过程。新的函数返回一个以原有第二个参数为参数的函数。Scala 偏应用函数是一种表达式，不需要提供函数需要的所有参数，只需要提供部分，或不提供所需参数。

Scala中的集合和Python差不多，Scala 集合分为可变的和不可变的集合。可变集合可以在适当的地方被更新或扩展，这意味着可以修改，添加，移除一个集合的元素。不可变集合不会改变，但仍然可以模拟添加，移除或更新操作，只是这些操作将在每一种情况下都返回一个新的集合，同时使原来的集合不发生改变。Scala中几种常用集合类型是：List(列表)，Set(集合)，Map(映射)，元组。

Scala 列表类似于数组，所有元素的类型都相同，列表是不可变的，值一旦被定义了就不能改变。可以使用 ::: 运算符或 List.concat()方法来合并两个或多个列表。使用 List.fill()方法来创建一个指定重复数量的元素列表。List.tabulate() 方法是通过给定的函数来创建列表。List.reverse 用于将列表的顺序反转。Scala Set集合中的对象不按特定的方式排序，并且没有重复对象，所有的元素都是唯一的。Scala Set集合分为可变的和不可变的集合。默认情况下Scala使用的是不可变集合，默认引用scala.collection.immutable.Set包，如果想使用可变集合，需要引用 scala.collection.mutable.Set包。Map 是一种把键对象和值对象映射的集合，它的每一个元素都包含一对键对象和值对象,所有的值都可以通过键来获取,Map 中的键都是唯一的,Map 也叫哈希表Hash tables。默认情况下Scala使用不可变 Map。如果需要使用可变集合，需要显式的引入import scala.collection.mutable.Map类,在Scala中可以同时使用可变与不可变 Map，不可变的直接使用 Map，可变的使用 mutable.Map。元组与列表类似，但与列表不同的是元组可以包含不同类型的元素。Scala Iterator（迭代器）不是一个集合，它是一种用于访问集合的方法。迭代器 it 的两个基本操作是 next 和 hasNext。调用 it.next() 会返回迭代器的下一个元素，并且更新迭代器的状态。调用 it.hasNext() 用于检测集合中是否还有元素。让迭代器 it 逐个返回所有元素最简单的方法是使用 while 循环。

Scala类是对象的抽象，而对象是类的具体实例。类是抽象的，而对象是具体的。类是用于创建对象的蓝图，它是一个定义包括在特定类型的对象中的方法和变量的软件模板。和Java或C++一样，Scala可以有任意多的构造器。除了主构造器之外，类还可以有任意多的辅助构造器。在Scala中，每个类都有主构造器。主构造器并不以this方法定义，而是与类定义交织在一起，主构造器的参数直接放置在类名之后 。内部类是在类中定义的类，Scala通过两种方式在内部类访问外部类成员。Scala没有静态方法或静态字段，可以用object这个语法结构来达到同样目的。在Java或C++中，通常会用到既有实例方法又有静态方法的类。在Scala中，可以通过类和与类同名的“伴生”对象来达到同样的目的。伴生对象方便类的构建，可做为当前类的静态方法、成员的集合。提取器是一个unapply方法的对象，可以把unapply方法当做是伴生对象中apply方法的反向操作。Apply方法接受构造参数，然后将它们变成对象，unapply方法接受一个对象，然后从中提取值，通常这些值就是当初用来构造该对象的值。unapply方法返回的是一个Option，它包含一个元组，每个匹配到的变量各有一个值与之对应。在Scala中，Trait是一种特殊概念，代表特征的意思。Trait可以被作为接口来使用，此时Trait与Java的接口非常类似。类使用extends继承Trait，与Java不同，这里不是implement，在Scala中，无论继承类还是继承Trait都是用extends关键字。在Scala中，类继承Trait后，必须实现其中的抽象方法。Scala同Java一样，不支持类的多继承，但支持多重继承Trait，使用with关键字即可。Scala 提供了强大的模式匹配机制，应用也非常广泛。一个模式匹配包含了一系列备选项，每个都开始于关键字 case。每个备选项都包含了一个模式及一到多个表达式。箭头符号 => 隔开了模式和表达式。

3. Spark学习总结

Spark是作为机器学习的核心部分来讲解的。Apache Spark是一个快速的，多用途的，分布式的计算框架，在内存中比 Hadoop MapReduce 快100倍，在硬盘上比它快10倍，并且支持多种语言（Java，Scala，Python，R），可以针对业务使用 SQL，流式处理或更复杂的逻辑规则，并且可以在 Hadoop，Mesos，Kubernetes 上运行。

Spark shell 提供了一种来学习该 API 比较简单的方式, 以及一个强大的来分析数据交互的工具。在 Scala（运行于 Java 虚拟机之上, 并能很好的调用已存在的 Java 类库）或者 Python 中它是可用的。通过在 Spark 目录中运行以下的命令来启动它: ./bin/spark-shell。Spark主要抽象是一个称为 Dataset 的分布式的 item 集合。Datasets 可以从 Hadoop 的 InputFormats（例如 HDFS文件）或者通过其它的 Datasets 转换来创建。让我们从 Spark 源目录中的 README 文件来创建一个新的 Dataset: 可以直接从 Dataset 中获取 values（值）, 通过调用一些 actions（动作）, 或者 transform（转换）Dataset 以获得一个新的。统计出现次数最多的行 : textFile.map(line => line.split(" ").size).reduce((a, b) => if (a > b) a else b)。

我们主要学习了RDD和spark的mllib库。 Resilient Distributed Datasets（RDD，） 弹性分布式数据集。RDD是只读的、分区记录的集合。RDD只能基于在稳定物理存储中的数据集和其他已有的RDD上执行确定性操作来创建。这些确定性操作称之为转换，如map、filter、groupBy、join（转换不是程开发人员在RDD上执行的操作）。RDD不需要物化。RDD含有如何从其他RDD衍生（即计算）出本RDD的相关信息（即Lineage），据此可以从物理存储的数据计算出相应的RDD分区。RDD支持两种操作：转换（transformation）从现有的数据集创建一个新的数据集；而动作（actions）在数据集上运行计算后，返回一个值给驱动程序。 例如，map就是一种转换，它将数据集每一个元素都传递给函数，并返回一个新的分布数据集表示结果。另一方面，reduce是一种动作，通过一些函数将所有的元素叠加起来，并将最终结果返回给Driver程序。（不过还有一个并行的reduceByKey，能返回一个分布式数据集）。

MLlib 是 Spark 的机器学习（ML）库。其目标是使实用的机器学习具有可扩展性并且变得容易。在较高的水平上，它提供了以下工具：ML Algorithms （ML 算法）: 常用的学习算法，如分类，回归，聚类和协同过滤；Featurization （特征）: 特征提取，变换，降维和选择；Pipelines （管道）: 用于构建，评估和调整 ML Pipelines 的工具；Persistence （持久性）: 保存和加载算法，模型和 Pipelines；Utilities （实用）: 线性代数，统计学，数据处理等。

Spark SQL 的功能之一是执行 SQL 查询.Spark SQL 也能够被用于从已存在的 Hive 环境中读取数据. 当以另外的编程语言运行SQL 时, 查询结果将以 Dataset/DataFrame的形式返回.也可以使用命令行或者通过 JDBC/ODBC与 SQL 接口交互.

Spark Streaming 是 Spark Core API 的扩展, 它支持弹性的, 高吞吐的, 容错的实时数据流的处理. 数据可以通过多种数据源获取, 例如 Kafka, Flume, Kinesis 以及 TCP sockets, 也可以通过例如 map, reduce, join, window 等的高级函数组成的复杂算法处理. 最终, 处理后的数据可以输出到文件系统, 数据库以及实时仪表盘中.还可以在 data streams（数据流）上使用机器学习以及图计算算法.



4. Hadoop学习总结

Apache Hadoop是一款支持数据密集型分布式应用程序并以Apache 2.0许可协议发布的开源软件框架。它支持在商品硬件构建的大型集群上运行的应用程序。Hadoop是根据谷歌公司发表的MapReduce和Google文件系统的论文自行实现而成。所有的Hadoop模块都有一个基本假设，即硬件故障是常见情况，应该由框架自动处理。

Hadoop框架透明地为应用提供可靠性和数据移动。它实现了名为MapReduce的编程范式：应用程序被分区成许多小部分，而每个部分都能在集群中的任意节点上运行或重新运行。此外，Hadoop还提供了分布式文件系统，用以存储所有计算节点的数据，这为整个集群带来了非常高的带宽。MapReduce和分布式文件系统的设计，使得整个框架能够自动处理节点故障。它使应用程序与成千上万的独立计算的计算机和PB级的数据连接起来。现在普遍认为整个Apache Hadoop“平台”包括Hadoop内核、MapReduce、Hadoop分布式文件系统（HDFS）以及一些相关项目，有Apache Hive和Apache HBase等等。

hive是基于Hadoop的一个数据仓库工具，可以将结构化的数据文件映射为一张数据库表，并提供简单的SQL查询功能，可以将SQL语句转换为MapReduce任务进行运行。其优点是学习成本低，可以通过类SQL语句快速实现简单的MapReduce统计，不必开发专门的MapReduce应用，十分适合数据仓库的统计分析。

三. 专题内容分析

这部分内容主要谈论一下自己在实习过程中做的项目.

在实习的项目阶段，我选择了个人做一个大数据的竞赛题目，我不想花时间到无聊的界面和已经做了很多次的数据库编程，我们组的成员每个人都选择了做一道关于大数据竞赛的题目，通过这种方法来学习一些大数据的知识。

我选择了DC大数据竞赛上的一道新手入门题目，预测美国King Country的房价。



数据主要包括2014年5月至2015年5月美国King County的房屋销售价格以及房屋的基本信息。数据分为训练数据和测试数据，分别保存在kc\_train.csv和kc\_test.csv两个文件中。其中训练数据主要包括10000条记录，14个字段，主要字段说明如下：

第一列“销售日期”：2014年5月到2015年5月房屋出售时的日期

第二列“销售价格”：房屋交易价格，单位为美元，是目标预测值

第三列“卧室数”：房屋中的卧室数目

第四列“浴室数”：房屋中的浴室数目

第五列“房屋面积”：房屋里的生活面积

第六列“停车面积”：停车坪的面积

第七列“楼层数”：房屋的楼层数

第八列“房屋评分”：King County房屋评分系统对房屋的总体评分

第九列“建筑面积”：除了地下室之外的房屋建筑面积

第十列“地下室面积”：地下室的面积

第十一列“建筑年份”：房屋建成的年份

第十二列“修复年份”：房屋上次修复的年份

第十三列"纬度"：房屋所在纬度

第十四列“经度”：房屋所在经度

对于这个题目，我采用了老师在上课时讲的一般流程来处理：数据清洗，特征筛选，训练数据划分为训练集和测试集，模型训练，参数调优，预测。

在数据清洗阶段，分三步进行：1.根据我们的直觉，会觉得房价与住房面积，地下室面积，排名，建筑年份有关，根据这样的直觉对房价与这些变量的关系进行逐一分析，但是需要强调，这只是我们的主观臆断. 2. 接下来我们应该冷静下来，做一些看起来更加智慧的工作。我们应该考虑所有的特征，进行地毯式搜索，不应该忽略我们看起来不重要的因素.3. 正态，同性.

在特征筛选阶段，我选择了2种方法来：包装法和过滤法。2种方法都是选择相关性最强的前8个特征属性，2种方法中过滤法更好一些。最后选出的前8个属性有7个都是重合的，只有最后一个不一样。

在数据划分阶段，我把数据划分为3:1，3用来训练数据，1用来对模型进行检测，看拟合的结果。

在模型训练和参数调优阶段，我采用4种模型：线性回归，随机森林，gradient boosting ，xgboost.对于参数调优，先是大幅度大步长，确定比较优的取值后再是小范围小步长。这是一般的参调优技巧。

最后利用获取的最优化参数，把所有的数据都用来训练模型。把最后三个模型进行加权平均，获取最后的预测模型，用来预测test文件。

下面主要对4种机器学习方法进行详细的介绍：

1. 线性回归

1）算法

　对于线性回归的损失函数J(θ)=1/2(Xθ−Y)T(Xθ−Y)，我们常用的有两种方法来求损失函数最小化时候的θ参数：一种是梯度下降法，一种是最小二乘法。

如果采用梯度下降法，则θ的迭代公式是这样的：θ=θ−αXT(Xθ−Y)。通过若干次迭代后，我们可以得到最终的θ的结果。如果采用最小二乘法，则θ的结果公式如下：θ=(XTX)−1XTY。当然线性回归，还有其他的常用算法，比如牛顿法和拟牛顿法。

2）推广

（1）多项式推广

回到我们开始的线性模型，hθ(x1,x2,...xn)=θ0+θ1x1+...+θnxn, 如果这里不仅仅是x的一次方，比如增加二次方，那么模型就变成了多项式回归。这里写一个只有两个特征的p次方多项式回归的模型：

hθ(x1,x2)=θ0+θ1x1+θ2x2+θ3x21+θ4x22+θ5x1x2

我们令x0=1,x1=x1,x2=x2,x3=x21,x4=x22,x5=x1x2 ,这样我们就得到了下式：

hθ(x1,x2)=θ0+θ1x1+θ2x2+θ3x3+θ4x4+θ5x5

可以发现，我们又重新回到了线性回归，这是一个五元线性回归，可以用线性回归的方法来完成算法。对于每个二元样本特征(x1,x2),我们得到一个五元样本特征(1,x1,x2,x21,x22,x1x2)，通过这个改进的五元样本特征，我们重新把不是线性回归的函数变回线性回归.

（2）广义线性回归

我们对于特征y做推广。比如我们的输出Y不满足和X的线性关系，但是lnY 和X满足线性关系，模型函数如下：lnY=Xθ

这样对与每个样本的输入y，我们用 lny去对应， 从而仍然可以用线性回归的算法去处理这个问题。我们把 Iny一般化，假设这个函数是单调可微函数g(.),则一般化的广义线性回归形式是：g(Y)=Xθ 或者 Y=g−1(Xθ)

2. 随机森林

理解了bagging算法，随机森林(Random Forest,以下简称RF)就好理解了。它是Bagging算法的进化版，也就是说，它的思想仍然是bagging,但是进行了独有的改进。我们现在就来看看RF算法改进了什么。首先，RF使用了CART决策树作为弱学习器，这让我们想到了梯度提示树GBDT。第二，在使用决策树的基础上，RF对决策树的建立做了改进，对于普通的决策树，我们会在节点上所有的n个样本特征中选择一个最优的特征来做决策树的左右子树划分，但是RF通过随机选择节点上的一部分样本特征，这个数字小于n，假设为nsub，然后在这些随机选择的nsub个样本特征中，选择一个最优的特征来做决策树的左右子树划分。这样进一步增强了模型的泛化能力。如果nsub=n，则此时RF的CART决策树和普通的CART决策树没有区别。nsub越小，则模型约健壮，当然此时对于训练集的拟合程度会变差。也就是说nsub越小，模型的方差会减小，但是偏倚会增大。在实际案例中，一般会通过交叉验证调参获取一个合适的nsub的值。

除了上面两点，RF和普通的bagging算法没有什么不同， 下面简单总结下RF的算法。输入为样本集D={(x,y1),(x2,y2),...(xm,ym)}，弱分类器迭代次数T。输出为最终的强分类器f(x)

1）对于t=1,2...,T:

a)对训练集进行第t次随机采样，共采集m次，得到包含m个样本的采样集Dt

　　　　b)用采样集Dt训练第t个决策树模型Gt(x)，在训练决策树模型的节点的时候， 在节点上所有的样本特征中选择一部分样本特征， 在这些随机选择的部分样本特征中选择一个最优的特征来做决策树的左右子树划分

2) 如果是分类算法预测，则T个弱学习器投出最多票数的类别或者类别之一为最终类别。如果是回归算法，T个弱学习器得到的回归结果进行算术平均得到的值为最终的模型输出。这个函数g(.)我们通常称为联系函数。

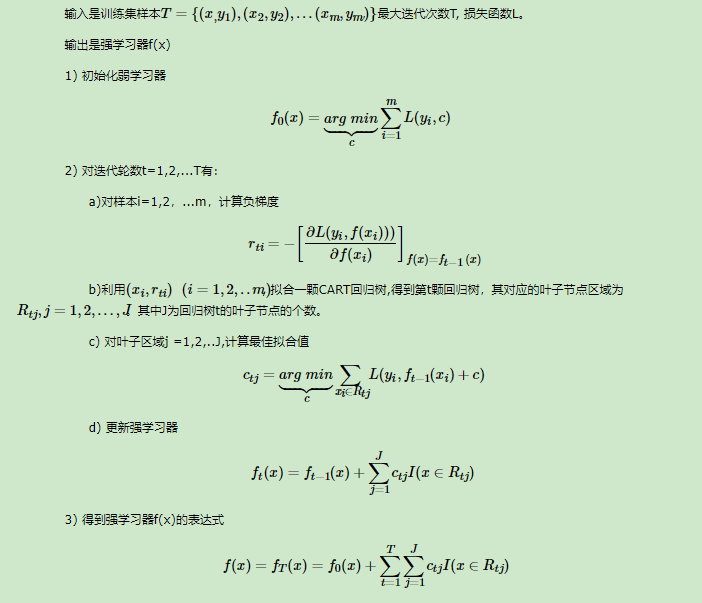
3. gradient boosting tree

1）概述

GBDT也是集成学习Boosting家族的成员，但是却和传统的Adaboost有很大的不同。回顾下Adaboost，我们是利用前一轮迭代弱学习器的误差率来更新训练集的权重，这样一轮轮的迭代下去。GBDT也是迭代，使用了前向分布算法，但是弱学习器限定了只能使用CART回归树模型，同时迭代思路和Adaboost也有所不同。在GBDT的迭代中，假设我们前一轮迭代得到的强学习器是ft−1(x), 损失函数是L(y,ft−1(x)), 我们本轮迭代的目标是找到一个CART回归树模型的弱学习器ht(x)，让本轮的损失损失L(y,ft(x)=L(y,ft−1(x)+ht(x))最小。也就是说，本轮迭代找到决策树，要让样本的损失尽量变得更小。

GBDT的思想可以用一个通俗的例子解释，假如有个人30岁，我们首先用20岁去拟合，发现损失有10岁，这时我们用6岁去拟合剩下的损失，发现差距还有4岁，第三轮我们用3岁拟合剩下的差距，差距就只有一岁了。如果我们的迭代轮数还没有完，可以继续迭代下面，每一轮迭代，拟合的岁数误差都会减小。

2）算法



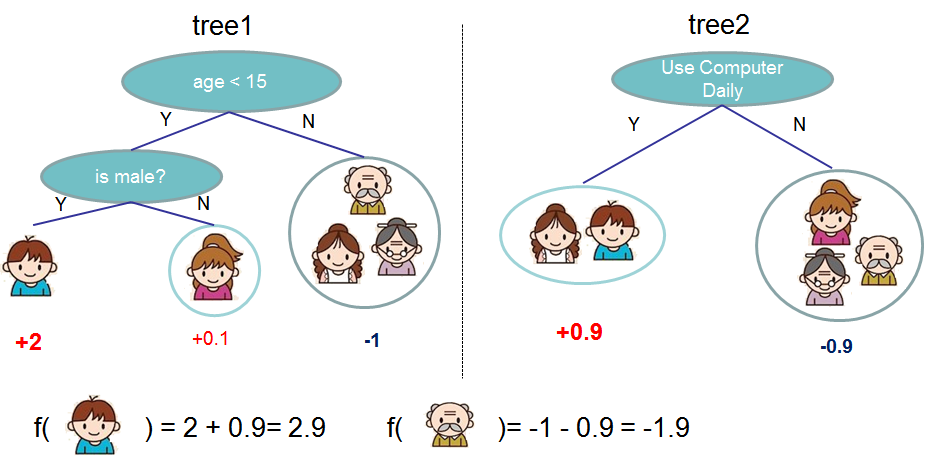
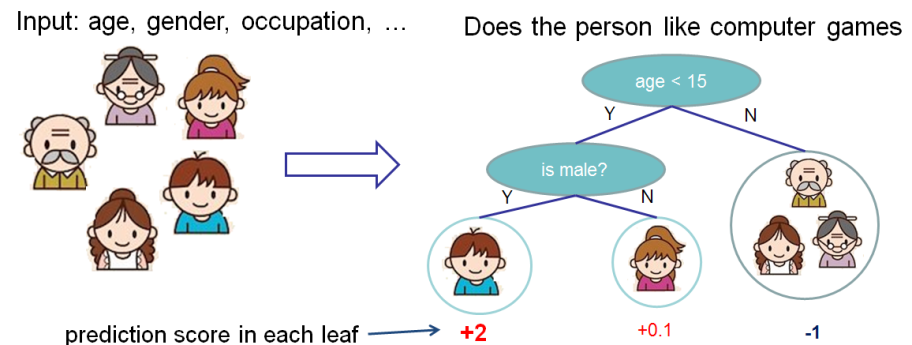
4. xgboost

XGBoost 是一个可伸缩且灵活的梯度提升框架，具有许多优秀的特性，它是灵活的（支持回归, 分类, 排序和用户定义的目标函数），可移植的（跨操作系统），多语言的（支持多种语言, 包括 C++, Python, R, Java, Scala），分布式的，高性能的，唯一的遗憾是暂未支持 GPU 的特性。

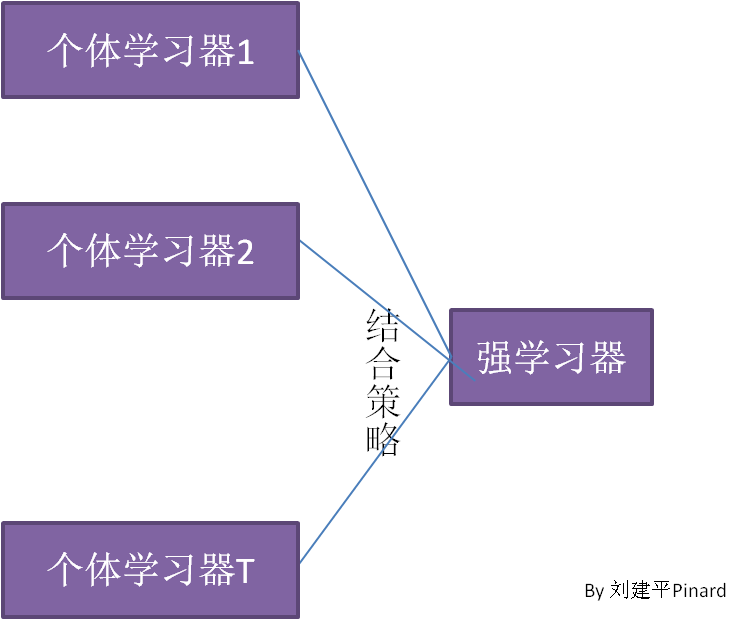
XGBoost 用于监督学习问题，我们使用训练数据 xi来预测目标变量 yi.

对于XGboost的算法原理，官网已经给出了详细的说明，在这里不再赘述。

只贴出一张原理图给出示例性的说明：



最后探讨一下集成学习的原理：



集成学习的结合策略有3种：

假定我得到的T个弱学习器是{h1,h2,...hT}

1. 平均法：

对于数值类的回归预测问题，通常使用的结合策略是平均法，也就是说，对于若干个弱学习器的输出进行平均得到最终的预测输出。

最简单的平均是算术平均，也就是说最终预测是H(x)=1T∑1Thi(x)

如果每个个体学习器有一个权重w，则最终预测是H(x)=∑i=1Twihi(x)其中wi是个体学习器hi的权重，通常有wi≥0,∑i=1Twi=1

2. 投票法

对于分类问题的预测，我们通常使用的是投票法。假设我们的预测类别是{c1,c2,...cK},对于任意一个预测样本x，我们的T个弱学习器的预测结果分别是(h1(x),h2(x)...hT(x))。

最简单的投票法是相对多数投票法，也就是我们常说的少数服从多数，也就是T个弱学习器的对样本x的预测结果中，数量最多的类别ci为最终的分类类别。如果不止一个类别获得最高票，则随机选择一个做最终类别。

稍微复杂的投票法是绝对多数投票法，也就是我们常说的要票过半数。在相对多数投票法的基础上，不光要求获得最高票，还要求票过半数。否则会拒绝预测。

更加复杂的是加权投票法，和加权平均法一样，每个弱学习器的分类票数要乘以一个权重，最终将各个类别的加权票数求和，最大的值对应的类别为最终类别。

3. 学习法

对于学习法，代表方法是stacking，当使用stacking的结合策略时， 我们不是对弱学习器的结果做简单的逻辑处理，而是再加上一层学习器，也就是说，我们将训练集弱学习器的学习结果作为输入，将训练集的输出作为输出，重新训练一个学习器来得到最终结果。

在这种情况下，我们将弱学习器称为初级学习器，将用于结合的学习器称为次级学习器。对于测试集，我们首先用初级学习器预测一次，得到次级学习器的输入样本，再用次级学习器预测一次，得到最终的预测结果。

四. 收获与体会

这次实习过程中最大的收获就是学习了一些机器学习的基础知识，利用编程对机器学习方法有了一个直观的理解，不再感觉到机器学习是一件神秘的事情。

前期对Scala的学习，对函数式编程有了一定的了解，很短的时间不能说自己学会了Scala的语法，但是Scala的面向对象和面向函数的特性给我留下了很深的印象，为以后学习编程范式起到了基础作用。第二周学习的spark和Hadoop相关的技术栈，扩展了我的视野，揭开了大数据和机器学习的冰山一角，虽然在Linux环境下配置这些框架曾经让我痛苦不堪，但是加深了我对Linux的理解，对Linux的终端命令有了进一步的学习。还记得第一次配置hadoop的时候花了一天的时间还没有配置好，第二天早上在官方文档的指导下半小时就配置好了，说明了学习最快和最好的方法就是去看官方文档，网上的一些博客是靠不住的。在学习的过程中，多和同学交流，不要闷着头自己一个劲的死钻，这样浪费时间还会丧失信心。我在配置框架的时候就得到了一些同学的帮助，很感谢他们。并不是一遇到问题就去找人问，学会Google是一个学计算机的学生的必备素质。接下来的一周的项目时间，我明白了公司软件开发的一般流程，对软件文档有了新的认识，以前自己写代码的时候很不喜欢写文档，感觉这是浪费时间，其实磨刀不误砍柴工，提前分析好需求，规划好算法，框架，技术，会为后续的编码工作减少大量的时间。在项目阶段，我选择个人做一道大数据的题目，虽然不是系统，但是难度并不低于推荐系统中推荐算法。在完成的过程中，我学习了很多东西，理论转换成了实践，对集成学习的几种算法有了深刻的理解。

五. 改进意见

一个月的实习时间说短不短，说长也不长，感谢东软老师的一路陪伴，让我学习了很多知识，这些知识在平时上课很难学到。

意见我很难提出来，我觉得老师讲的不错，也很注重和我们互动，对我们的困难能及时解决，大数据的老师真心不错。