Spark MLIB

1. 机器学习
   1. 什么是机器学习

机器学习(Machine Learning，ML)是一门多领域的交叉学科，涉及概率论、统计学、逼近论、算法复杂度理论等多门学科。专门研究计算机怎样模拟或实现人类的学习行为，以获取新的知识和技能，重新组织已有的数据结构使之不断改善自身的性能。简单说就是通过算法使计算机能够模拟人类的判断能力执行不同的操作。

机器学习是人工智能的核心，是计算机具有智能的根本途径，其应用遍及人工智能的各个领域。

* 1. 机器学习的应用

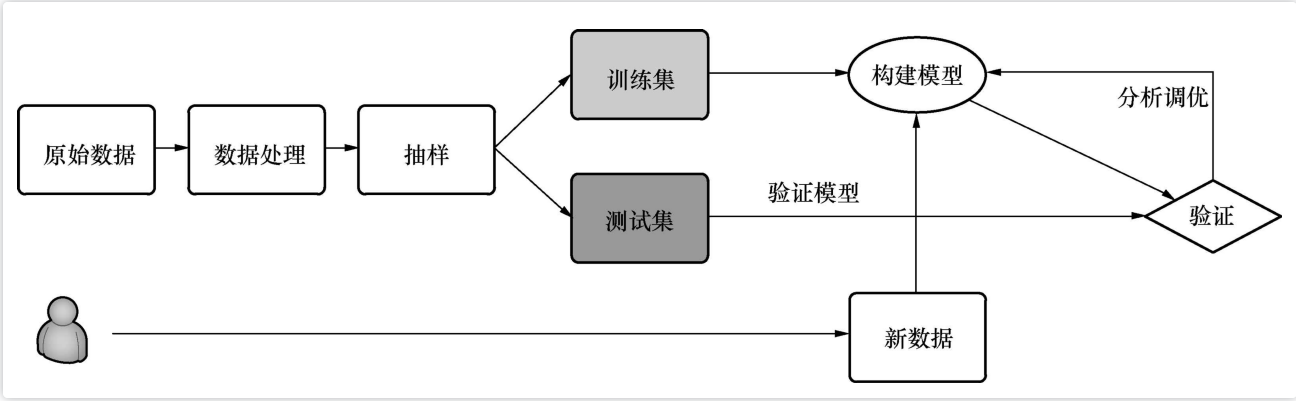


* 1. 机器学习中的几个分类
* **监督学习(supervised learning)**

监督学习是在机器学习过程中提供对错提示，一般在训练的数据集中包含最后结果，通过算法让机器自我减少误差。这类学习主要应用于分类和预测(regression classify)。监督学习从给定的训练数据集中学习出一个函数，当新的数据到来时，可以根据这个函数预测结果。监督学习的训练集要求包括输入和输出，也可以说是特征和目标。训练集中的目标是由人标注的。常见的监督学习算法包括回归分析和统计分析等。

例如：朴素贝叶斯、SVM、随机森林、逻辑回归、线性回归

监督学习流程如下图所示：

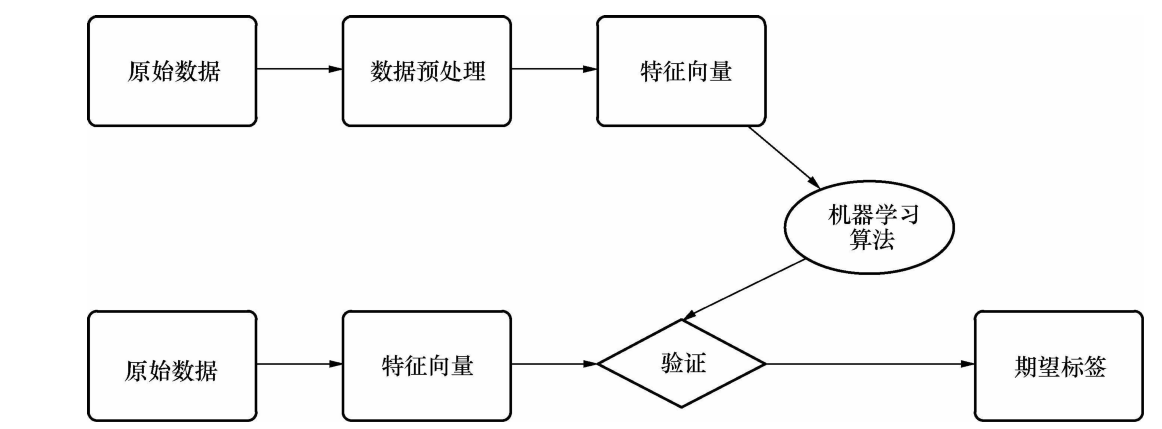


从图中可以看出，经过数据预处理后，数据被分为两部分：一部分是测试集，另一部分是训练集。通过学习算法，可以由训练集得到所需的模型，在通过验证集进行验证，最后根据验证结果分析调优，完善优化数据模型。

* **无监督学习（unsupervised learning）**

无监督学习又称为归纳学习(clustering)利用K方式(Kmeans)，建立中心（centriole）,通过循环和递减运算来减小误差，达到分类的目的。

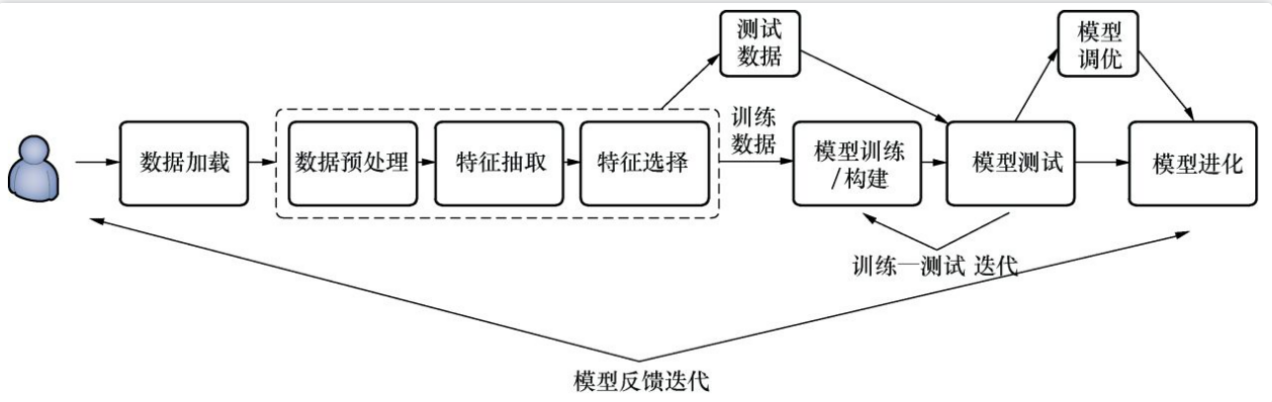
例如：K-means聚类、主成分分析、SVD



无监督学习过程中，通常使用聚类技术，基于某些相似性度量来对未标记样本进行分组。因此无监督学习任务会涉及挖掘隐藏的模式、特征学习等。聚类是智能地对数据集中的元素进行分类的过程。总体思路是同一类中的两个元素比属于不同类中的元素彼此更为“接近”。

* 1. 机器学习怎么用

典型的机器学习应用程序涉及以下步骤:



1. 线性回归

1）创建Maven项目

在IDEA中创建Maven项目，修改pom.xml文件添加项目MLIB依赖包，关键代码如下所示：

|  |
| --- |
| <dependency>  <groupId>org.apache.spark</groupId>  <artifactId>spark-mllib\_2.12</artifactId>  <version>${spark.version}</version>  </dependency> |

1. 线性回归关键代码

|  |
| --- |
| package com.software.client  import org.apache.spark.ml.regression.LinearRegression  import org.apache.log4j.{Level, Logger}  import org.apache.spark.SparkConf  import org.apache.spark.rdd.RDD  import org.apache.spark.sql.{DataFrame, Dataset, Row, SparkSession}  import org.apache.spark.ml.feature.{OneHotEncoder, StringIndexer}  import org.apache.spark.sql.functions.expr  import org.apache.spark.ml.feature.VectorAssembler  import org.apache.spark.ml.regression.LinearRegression  import org.apache.spark.sql.types.{DoubleType, IntegerType, StringType, StructField, StructType}  import org.apache.spark.sql.functions.udf  import scala.collection.immutable  /\*\*  \* 机器学习  \*/  object MachineWeatherExample {  Logger.getLogger("org").setLevel(Level.ERROR)  def main(args: Array[String]): Unit = {  //1.创建Spark环境配置对象  val conf = new SparkConf().setAppName("MachineWeatherExample").setMaster("local")  //2.创建SparkSession对象  val spark:SparkSession = SparkSession.builder().config(conf).getOrCreate()  var data = spark.read.option("delimiter", ",")  .option("header",true)  .option("multiLine", true)  .csv("input/countryTemperatures.csv")  import spark.implicits.\_  import org.apache.spark.sql.functions.\_  //1.数据处理 正则表达式的替换  data=regexpReplace(data,"dt","/","-")  data.show(5)  //2.数据清洗 删除不是空值的记录  val filteredData = data.filter(col("dt").isNotNull  && col("temperature").isNotNull  && col("country").isNotNull  && !col("country").like("%,%")  && !col("country").like("%?%"))  //3.数据加工  //3.1 数据格式转换  filteredData.withColumn("dt",(to\_date(col("dt"),"yyyy-MM")))  //3.2 日期按年、月拆分进行转换  val featureData = filteredData.withColumn("year",year($"dt"))  .withColumn("month",month($"dt"))  .select("year","month","country","temperature")  //featureData.show(10)  //3.2 自然语言使用StringIndexer进行标签编码  val indexer = new StringIndexer()  .setInputCol("country")  .setOutputCol("countryIndex")  //3.3 拟合StringIndexer模型，生成编码器  val codemodel = indexer.fit(featureData)  val datamodel =codemodel.transform(featureData)  //datamodel.show(20)  //3.4 生成新的数据集  val result = datamodel.map(  row=>(row.getInt(0),row.getInt(1),row.getDouble(4),row.getString(3).toDouble)  ).toDF("year","month","countryIndex","temperature")  //result.show(10)  //3.5 数据预处理  // 创建了一个 VectorAssembler 对象，并设置了输入列和输出列。setInputCols 方法接收一个字符串数组，  // 参数是要组合成向量的列名，这里是 "year"、"month" 和 "countryIndex"。  // setOutputCol 设置了输出列的名称为 "features"，该列将包含组合后的向量特征。  val assembler = new VectorAssembler()  .setInputCols(Array("year", "month", "countryIndex")) // 填写实际的日期特征列名  .setOutputCol("features")  //使用 transform 方法将 result中的列数组转换为向量列。transform方法接收一个输入DataFrame，  //并根据之前设置的输入和输出列进行转换。转换后的结果保存在新的 DataFrame中，这里命名为assembledData。  //然后，使用select方法选择了转换后的features列和原始的temperature列，生成最终的 DataFrame。  val assembledData = assembler.transform(result).select("features", "temperature")  //4.1拆分数据集为训练集和测试集（按比例拆分）  val Array(trainingData, testData) = assembledData.randomSplit(Array(0.7, 0.3))  trainingData.show(10)  //4.2 创建线性回归模型  val lr = new LinearRegression()  .setLabelCol("temperature")  .setFeaturesCol("features")  //模型训练  val model = lr.fit(trainingData)  // 测试集进行预测  val predictions = model.transform(testData)  // 打印预测结果  predictions.select("prediction", "temperature", "features").show()  // 关闭SparkSession  spark.stop()  }  //正则替换  def regexpReplace(df:DataFrame,columnName:String,regexp:String,newValue:Any):DataFrame={  val exprString:String = "regexp\_replace("+columnName+",'"+regexp+"','"+newValue+"')"  df.withColumn(columnName,expr(exprString).alias(columnName))  }  } |