Introduce

//todo

预备工作

// todo

学习索引模块

//todo

自动选择模型：

对于一个指定的查询，学习索引模块已经实现了逐层缩减查询位置的相关字段的大小，给出了尽可能精确的位置信息。然而，仍有两个问题需要解决：1. 给出的结果仍然是范围的，需要明确到一个唯一的位置。2. 给出的结果是基于误差上界的，我们需要保证通过本文的架构所查询到的结果，就是表中存储的结果。为了解决这两个问题，就需要在底层加入适当的回归模型，进一步拟合，缩减相关字段的范围，得到精确解。然后对于解进行讨论与处理，保证查询得到的数据的真实性。

我们首先提出个体差异性强，覆盖度较高的多种回归模型：m1,m2……mn。一方面，所谓个体差异性强，指的是这n个模型采取的机器学习方法不同，其中包括线性回归模型，逻辑回归模型，弹性回归模型，加入了平方偏差因子（正则项）的岭回归模型，基于集成学习的梯度下降树模型（GBDT）和极限树模型（ExtraTree）等。虽然都是回归模型，但是实现的技术已经涵盖了目前学习性能较好的大部分策略和方法。

另一方面，所谓覆盖度高，指的是我们使用的这些模型，考虑到了多种多样的数据来源，通过查询得到的数据中潜在的问题，以及数据内部潜在的关系：

数据可能出现过拟合的问题：集成学习的模型如GBDT和ExtraTree拟合过程中参数较多，容易发生过拟合行为。这个一方面可以通过调节模型参数，灵活选择，另一方面可以采取其他的比如多项式回归等，通过打分的方式，选出更优的模型。然而实际上，过拟合可以进一步提高我们查询结果的精确程度，因而只要在一定约束条件下，（例如当输出是字段数组下标等，要求是整数）这种过拟合现象反而可以使得我们模型效果更好。

数据内部可能存在函数依赖：在数据表中，数据的不同属性之间可能存在函数关系。比如在进行记账汇总操作时，对于表中的A1,A2……Am这m个属性，最后一个属性Am可能是前n-1个数据的累加和。数据内部的这种联系，使得在进行训练时，特征变量之间存在很高的共线性（high collinearity）。对于常规的线性回归与多项式回归，这会造成性能的严重下降。因此，我们在普通模型的优化函数的基础上，加入了正则项：一个较小的平方偏差因子：。这种方法会向模型中引入少量偏差，但是可以大幅减少方差。岭回归的公式如下所示：

由公式可知，当系数趋于0时，该目标函数就退化为普通的线性回归函数；当lambda区域正无穷，目标函数无穷大。因此为了使得目标函数达到最小，需要修改lambda的数值，使得正则项趋于0.因此，对于目标函数进行求导，得到如下公式：

，

为使得目标函数最小，当导函数为0时，目标函数取得极小值，此时取值如下所示：

可以看到，越大，方差越小，偏差越大。为了权衡方差与偏差的关系，我们采取k交查验证的方法。将数据集拆分为k个样本量大体相当的数据组并且没有重叠，从k个数据组中挑选k-1组数据用于模型的训练，剩下的一组用于模型测试，这将会得到k种训练集和测试集。每一种train和test，都会对应一个模型和模型的MSE（均方差）得分。构建岭回归模型的k重交叉验证，对于每个都会得到k个模型对应的得分，从而实现交叉验证。然后利用按照上述方法挑选出的构建岭回归模型即可。

提出的模型需要根据数据进行筛选，选择出当前table中，针对不同查询数据的最优模型。我们对于得到的数据集进行了如下算法1的处理：

**Input: X for the array of queried key, and y for the array of**

**the minimum range of predicted position**

**Output: X with new label**

**// give a list of models**

**modellist = {m1,m2,……,mn}**

**// initialize the parameters**

**max = 0, label = 0**

**while(i<modellist.size()):**

**//Compute score\_i using model\_i**

**scorei = mi(X,y).score()**

**if(scorei>max):**

**max = scorei**

**label = i**

**// add the chosen best model to X**

**addlabel(X,label)**

**return X**

**算法1：处理源查询数据X**

如算法一所示：该算法输入和输出是衔接与学习索引模块rmi的输出的。数据片X为用户查询的query命令中的值的数组；y是表中存储这精确位置的相关字段的数组，存储的是表中的索引。输出为增加了新标签的X，其中增加部分的是为数据片X选择的最优模型序号。需要注意的是，此算法输入的X是原来的X的切片，这样做可以使得：对同一个数据源的不同key的查询，可以利用并选择不同的模型。接下来要对于相同的一段数据，分别用modellist中模型**model\_i**进行训练，然后根据获得的参数，调用model类中的score方法，计算每个模型训练后的得分**score\_i，**并不断更新最大得分以及相应的标签记录label。循环结束后，对于当前输入的X，打上标签label，标志着这段数据应该选取modellist[label]进行相关字段的拟合效果最好。我们采取的评估标准包括SSE，MSE等，通过加入正则项和，优化我们的打分函数，得到如下公式：

其中，k为整体样本数量，代表y的预测值，即，经过模型拟合之后得到的索引位置。这里需要保证输出的y是整数，这一点我们在模型的输出接口中进行了处理。y对应于算法中的输入y，正则项用于尽可能消除高共线性和数据间的依赖，与是两个指标的权重占比，y是评估值的平均值。

我们注意到，这整个过程需要将同一个数据片经过modellist中所有模型的训练，涉及到整个的大小。如果等待一个数据集整体，完全跑完，后续处理工作只能待机，这就消耗了大量的时间与内存占用，并且效率较低。最理想的状态是，对于一个训练好的模型，只能由一个模型，接收用户查询的key的输入，用于拟合精确字段位置。所以，我们需要构建一个用于实现自动选择模型的分类器。此时，我们刚刚新标注了的数据X就派上了用场。我们以数据所选取的最优模型标号最为特征，采取随机森林分类器进行训练。选取随机森林分类器的原因是：待处理的数据具有维度高，特征复杂难抽取，并且可能具有噪声等问题。而随机森林的架构，决定了这种模型处理面向数据库的，复杂，带有噪声数据的天然优势。分类器分为训练使用和查询使用。训练后，对于一段数据片X，就会有特定的模型model\_i被选择进行拟合。最终输出结果是model拟合的结果，代码如下所示：

Input : X for the newly labeled X with chosen model order

y for the array of the minimum range of predicted position

Output : predicted index in the related field

// Get trainset and testset

if(mode==train):

Split trainset and testset for X

model = randomforest\_model(Parameters like n\_estimators,max\_depth,min\_samples\_split,……)

model.fit(X,y)

predict = model.predict(X\_train)

if(mode==query):

Load the local trained model

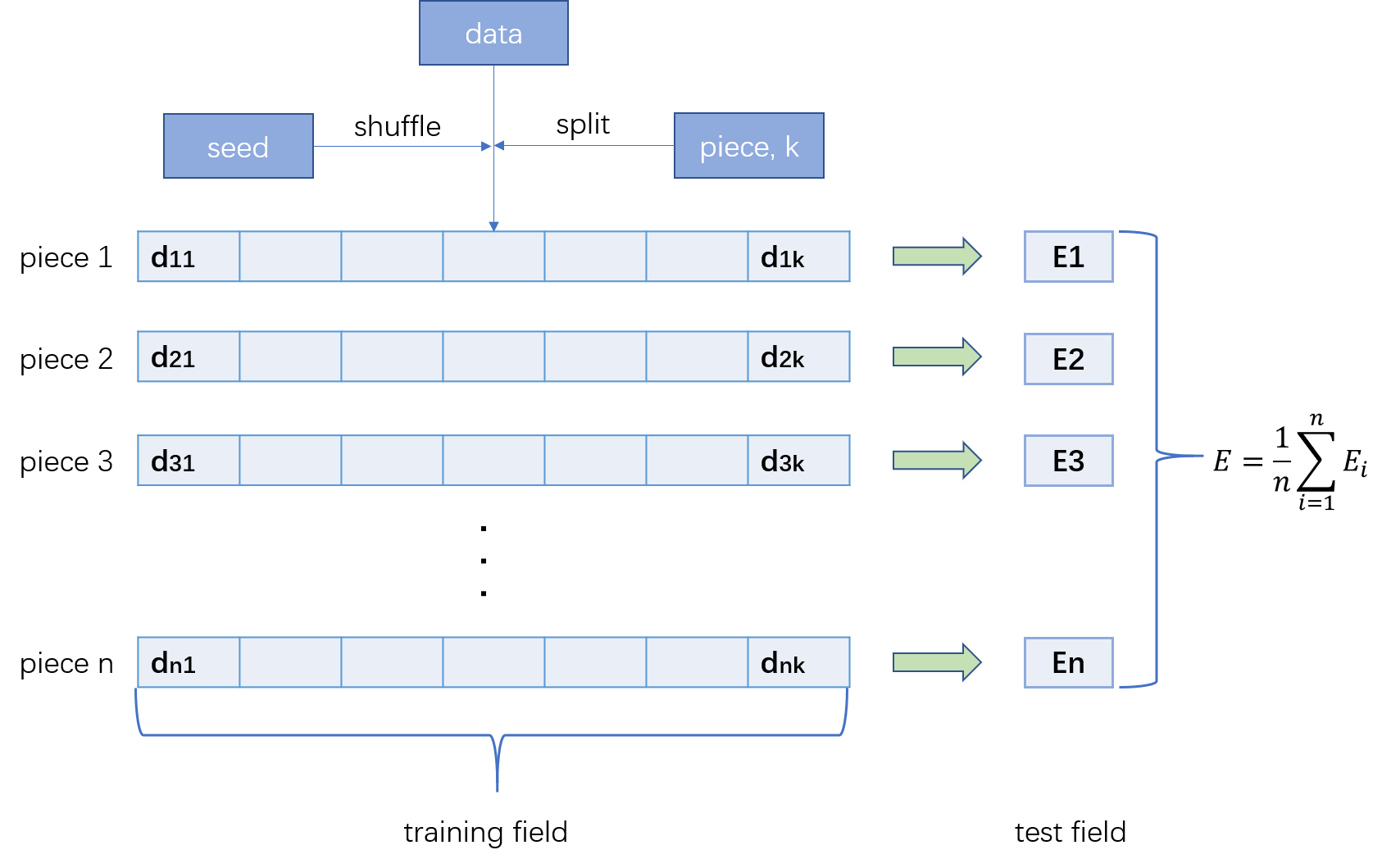
predict = model.predict(X)

return predict

对这个算法中的数据说明：由于对于同一数据源的单一表中的数据，进行分片操作后，首先进入算法1进行训练，标注标签，然后进入分类器进行分类，将选取相同类型的数据篇X放在一起。

然而，这样操作对于同一数据源的单一表仅有一轮，训练数据和轮次太少。因此，我们采取了如下的策略，用来增加训练的数据：对于每一轮的数据，首先用参数piece进行分割，分割后为了保证每轮训练的数据不同，我们将数据进行打乱操作，使其重分布。其中打乱（shuffle）操作采取人为选取的seed，每轮重分布结束后，seed增加，直到达到阈值，停止重分布生成数据。重分布生成的数据，经过k折交叉验证，直接进入算法一中进行标签标注，然后和之前一样，进入分类器进行训练。这样可以节省因为重分布生成新数据而占用额外的内存空间。

下图为经过参数seed，piece，k处理后的数据，在k折交叉验证下获取结果的方式。



由此，我们实现了模型的自动选择功能。总体来看，每当数据查询任务到来，首先经过学习索引模块，将sql语句转为key，定位到相关字段，进而通过学习索引的方式，逐步缩小范围到底层。底层通过构建训练好的多个回归模型以及分类器，让查询输入直接进入分类器，获取最优的回归模型，然后数据进入回归模型，找到在最小相关字段中的精确索引位置，根据输出情况，判断是否超出忍耐阈值，若超出则回表核对，否则直接输出即可。