目前可以用于分类问题的算法有许多种，比如：支持向量机、KNN法、Logistic回归、决策树算法以及一些集成类算法等。这些算法各有特点，可以确定的是没有任何一种算法可以做到在任何情况下都比其他算法表现的好，因为机器学习中的“没有免费的午餐定理”指出，没有任何一种算法对所有问题都有效，在监督学习（即预测建模）中尤其如此。本研究也对此深信不疑，所以本研究过程中，采用stacking方法结合了多种模型，以增强模型的稳定性和泛化性能，具体实现过程如下图。

K最邻近算法

随机森林

Xgboost

Logistic回归

训练样本

最终预测结果

训练样本作为输入

模型输出结果作为输入

本研究的建模思路大致如该图，首先以K最邻近算法、随机森林和xgboost作为初级学习器，以训练样本作为输入训练这三个模型，之后以初级学习器的输出作为次级学习器logistic回归的输入得到最终的预测结果。关于具体的建模过程和细节，请见下文。

1. **传统方法**

传统算法多种多样，目前来看违约预测方面多使用的分类模型算法，主要包括较为简便易理解的Logistic回归方法、决策树法等统计模型，也包括最近十分火热的神经网络等非统计模型。下面简要介绍一下几个较为传统的分类模型：

Logistic回归方法：Logistic回归方法是一类比较常见的信用评估方法，其属于广义线性回归，与多变量线性回归有一定的相似之处，它将数据拟合到一个Logit函数，从而能够完成对事件发生概率进行预测，通过设定一个阈值，即可实现对样本的分类。Logistic回归方法把信贷用户的相关数据作为输入，通过Logit函数，产生一个[0.0,1.0]之间的一个实数，其对应于用户可能违约的概率，只需设定一个临界值，便可区分用户类别。

决策树法：决策树是一类常见的机器学习算法，决策树是一种树形结构，其中每个内部节点表示一个属性上的测试，每个分支代表一个测试输出，每个叶节点代表一种类别。它是一种监管学习，所谓监管学习就是给定一堆样本，每个样本都有一组属性和一个类别，这些类别是事先确定的，那么通过学习得到一个分类器，这个分类器能够对新出现的对象给出正确的分类。利用决策树对信贷用户进行违约预测时，通过输入用户的一系列相关数据，从决策树根节点开始，在每个节点对用户不断进行判断，直至用户落入某个叶分支中，即可实现对用户类别的判断。

K最近邻方法：K最近邻方法的核心思想是如果一个样本在特征空间中的k个最相邻的样本中的大多数属于某一个类别，则该样本也属于这个类别，并具有这个类别上样本的特性。该方法在确定分类决策上只依据最邻近的一个或者几个样本的类别来决定待分样本所属的类别。该方法的思想比较简单，即特征空间中，将未知类别的用户与已知类别的用户的距离做比较，选取最邻近的一个或者几个用户的类别来决定未知用户的类别。

Logistic回归方法的简单易行，便于理解，可解释性较强，因此本研究从传统方法中选择了Logistic回归，其实现流程如下：

输入:数据集，阈值(终止误差)

输出：逻辑回归函数

流程：

* 1. 损失函数，其中，选取初始点
  2. 求
* 2.1 在第步，确定搜索方向
* 2.2 判断精度，若，则停止计算，否则转向2.3
* 2.3 确定最优步长
* 2.4 求出新点
* 2.5 重复2.1—2.4，直到停止运算，此时将结果记为–
  1. 得到

同时为了保证初级学习器具有较大的多样性，本次研究也选取了传统方法中的K最邻近算法，其实现流程如下：

* 1. 计算测试数据与各个训练数据之间的距离
  2. 按照距离的递增关系进行排序
  3. 选取距离最小的K个点
  4. 确定前K个点所在类别的出现频率
  5. 返回前K个点中出现频率最高的类别作为测试数据的预测分类

1. **集成算法**

集成学习通过将多个学习器进行结合，常可获得比单一学习器显著优越的泛化性能。这对“弱学习器”尤为明显，因此集成学习的很多理论研究都是针对弱学习器而言的，虽然从理论上来说，集成学习主要针对弱学习器，但根据实践和经验来看，经常是强学习器被用于集成。其中若希望集成学习获得比较优越的性能，其对基学习器也有一定的要求：基学习器既要有较高的准确性，又要保证基学习器有较大的差异性。集成学习大致可以分为两大类：bagging方法和boosting方法。

**Bagging方法**

Bagging是一种在原始数据集上通过有放回抽样重新选出和原来样本一样多新数据集来训练分类器的集成技术。也就是说这些新数据集是允许重复的。使用训练出来的分类器集合来对新样本进行分类，然后用多数投票（在一系列分类边界的中属于哪一类多就最终判定属于哪一类）或者对输出求均值的方法统计所有分类器的分类结果，结果最高的类别即为最终标签。

算法步骤：

1.从数据集中取样（放回选样），总共执行次

2.针对每一次取样训练得到分类模型，最终得到个模型

3.对未知样本X分类时,每个模型都得出一个分类结果，得票最高的即为未知样本X的分类

4.也可通过得票的平均值用于连续值的预测

其中bagging算法中比较有名的为随机森林算法，正式介绍随机森林算法之前，首先看一下信息增益和分类决策树的算法流程。

**信息增益**

输入：数据集和属性，表示第类样本的个数，即表示的样本的个数，表示的样本的个数

输出：属性对数据集的信息增益

流程：

1. 计算信息熵
2. 计算特征对数据集划分后的信息商，其中表示对应属性值取的样本的个数
3. 就算信息增益

**分类决策树**

输入：数据集和特征集,阈值

输出：决策树

流程：

1. 如果中样本全属于同一类别，则为单节点树，返回
2. 如果，将D中实例数最大的类作为该节点的类标记则为单节点树，返回
3. 否则按信息增益算法计算中每个特征对的信息增益，选择信息增益最大的特征
4. 如果 的信息增益，则置为单节点树，并将中实例数最大的类作为该节点的类标记，返回
5. 否则，对的每一个可能值,依将分割成若干非空子集，将中实例最大的类作为标记，构建子节点，由节点及其子节点构成树，返回
6. 对第个子节点，以为训练集，以为特征集，递归调用步，得到子树，返回

以上介绍了随机森林的基学习器分类决策树，接下来了解一下随机森林，随机森林算法和一般bagging方法基本流程相似，但是随机森林在建立基学习模型时，并不会选择所有的属性，其会在属性中随机抽取k个属性，这个做法使得随机森林的每个基学习模型有了更大的变异性，效果也更优，其算法过程如下：

输入:数据集，属性集，基学习算法，训练轮数，每棵树选择的属性个数，为类别集合

输出：随机森林

流程：

* 1. for 𝑡=1,2,⋯,𝑇
* 1.1 从属性集中随机选取个属性
* 1.2 利用自助法随机选取m个只包含相应属性的样本，记作
* 1.3
* end for

**Boosting方法**

Boosting是一族可将弱学习器提升为强学习器的算法。这族算法的工作机制类似：先从初始训练集训练出一个基学习器，再根据基学习器的表现对训练样本分布进行调整，使得先前基学习器做错的训练样本在后续受到更多关注，然后给予调整后的样本分布来训练下一个基学习器。如此重复进行，直至基学习器数目达到事先指定的值，最终将这个基学习器加权结合。本文所采用的boosting算法为xgboost，在正式介绍xgboost之前，首先看一下xgboost的基学习器：回归决策树。

**回归决策树**

输入：数据集

输出:树

流程：

1. 选择最优变量与切分点，求解，遍历变量，对固定的变量扫描切分点，选择使上式达到最小值的组合。
2. 用选定的划分区域并决定相应的输出值：

* 其中为结点中样本的平均值。

1. 继续对两个子区域调用步骤12，直到满足停止条件。
2. 将输入空间划分为个区域，生成决策树：，即当使用样本进行预测时，样本最终落入的叶子结点的均值作为最终的预测值。

接下来正式介绍xgboost模型，xgboost模型的数学表示其中表示树的棵数，表示所有可能的树，表示一颗具体的树。目标函数为，其中第一部分是损失函数，第二部分是正则项。

xgboost是加法训练，目标不是直接优化整个目标函数，而是首先优化第一棵树，之后再优化第二棵树，直至优化完K棵树。整个过程如下图所示：

在步时，添加了一棵最优的树，其就是在现有的棵树的基础上，使得目标函数最小的那棵树，此时目标函数可以写为，其中为棵树的复杂度。

对于一般损失函数，对其做泰勒二阶展开，如下所示：

其中的常数项不影响下面的分析，故删去得

对于其中的可以写作,其中代表树的叶子节点数，个叶子结点的值组成了一维向量，是一个映射，用来将样本映射成到的某个值，代表了树的结构。代表这棵树对样本的预测值。于是可以定义如下的正则化项，其中和为自定义值，二者越大，得到的树的结构越简单。

于是得到下式，其中，，指第棵树第个叶子结点上的样本的集合。

令对求导即可得叶子结点最佳值以及目标函数的值，即

其中表示了树的结构的优劣，其值越小，代表树的结构越好，同时可以注意到与第棵树的无关。

通过上面的分析可以发现，只要找到树的结构就可以找到叶子结点的值，也即求出了。对于寻找树的结构的问题，可以理解为是否根据某个属性切分数据来给原来的树增加一个结点，衡量标准为，其中表示属于新增的结点左半部分数据对应的值，表示属于新增节点右半部分数据对应的值。这个值为正且越大，表明增加一个节点后，树的结构越优，越值得增加一个结点。

1. **Stacking方法**

当训练数据很多时，一种更为强大的结合策略是使用”学习法”，即通过另一个学习器来进行结合。Stacking方法便是其中非常有名的一种方法，其流程如下：

输入：数据集

初级学习算法

次级学习算法

输出：集成算法

流程：

* 1. for

end for

* 1. 设
  2. for
* for
* end for
* end for

在训练阶段，次级训练集是利用初级学习器产生的，若直接利用初级学习器的训练集来产生次级训练集，则过拟合的风险会比较大；因此一般是通过使用交叉验证或者留一法这样的方式，用训练初级学习器未使用的样本来产生次级学习器的训练样本。以折交叉验证为例，初始训练集被随机划分为个大小相似的集合。令和分别表示第折的测试集和训练集。给定个初级学习算法，初级学习器通过在使用第学习算法而得。对中的每个样本，令，则由所产生的次级训练样本示例部分为，标记部分为。于是，在整个交叉验证过后，从这个初级学习器产生的次级训练集是，将用于训练次级学习器，流程如下图所示：

