北京科技大学
University of Science and Technology Beijing

密级：公开

本科生毕业设计(论文)

题 目：实验室地震来临时间的预测

作 者：许旭粮

学 号：41521208

学 院：数理学院

专 业：数学与应用数学

成 绩：

2019 年 06 月

本科生毕业设计(论文)

题　　目: 实验室地震来临时间的预测

英文题目: Time-Remaining Prediction of Laboratory Erathquakes

学　　院: 数理学院

班　　级: 数学 1501

学　　生: 许旭粮

学　　号: 41521208

指导教师: 储继迅　　**职称:** 副教授

指导教师:　_____职称：_____

声　　明

本人郑重声明：所呈交的论文是本人在指导教师的指导下进行的研究工作及取得研究结果。论文在引用他人已经发表或撰写的研究成果时，已经作了明确的标识；除此之外，论文中不包括其他人已经发表或撰写的研究成果，均为独立完成。其它同志对本文所做的任何贡献均已在论文中做了明确的说明并表达了谢意。

学生签名：_____　　_____年___月___日

导师签名：_____　　_____年___月___日

毕 业 设 计（论 文）任 务 书

一、学生姓名： 许旭粮　　学号：41521208

二、题　　目： 实验室地震来临时间的预测

三、题目来源： 自拟

四、结业方式： 论文

五、主要内容：

1. 学会并掌握聚类，回归，关联性分析，深度学习，时间序列分析，特征选择等数据分析方法的实现原理和应用方式；
2. 用上述方法对数据进行预测，对不同模型的表现进行评价分析；
3. 尝试对现有模型进行改进，并对结果进行总结分析.

六、主要(技术)要求：

1. 掌握聚类，回归，关联性分析，时间序列分析等算法和相应模型；
2. 分析，对比不同模型的优劣(或效率)；
3. 改进已有的分析方法和算法，给出实际预测，并对结果进行分析讨论；
4. 掌握 python 相关工具包，并编程实现.

七、日程安排：

1－ 2 周 查阅并学习文献，写出文献综述；
3－ 4 周 完成并参加选题报告；
5－ 8 周 完成多种数据方法的学习和应用评价，完成英文文献的翻译，填写中期检查表；
9－13 周 对现有模型进行改进进行预测，对结果进行评价分析，写出毕业论文；
14－15 周 论文送审并参加答辩.

八、主要参考文献和书目：

[1] Rouet-Leduc B , Hulbert C , Lubbers N , etc. Machine Learning Predicts Laboratory Earthquakes[J]. Geophysical Research Letters, 2017, 44(18):9276-9282.

[2] Rouet-Leduc B , Hulbert C , Bolton D C , etc. Estimating Fault Friction From Seismic Signals in the Laboratory[J]. Geophysical Research Letters, 2018, 45(3):1321-1329.

[3] Leeman J R , Saffer D M , Scuderi M M , etc. Laboratory observations of slow earthquakes and the spectrum of tectonic fault slip modes[J]. Nature Communications, 2016, 7:11104.

[4] Hulbert C , Rouet-Leduc B , Paul A.Johnson , etc. Similarity of fast and slow earthquakes illuminated by machine learning[J]. Nature Geoscience, 2019, 12:69-74.

[5] Rouet-Leduc B , Hulbert C , Paul A.Johnson. Continuous chatter of the Cascadia subduction zone revealed by machine learning[J]. Nature Geoscience, 2019, 12:75-79.

指导教师签字：　　年　月　日

学 生 签 字：　　年　月　日

系(所)负责人章：　　年　月　日

摘 要

预测地震的大小，位置和来临时间一直以来都是地质科学家们最基本的学术目标之一。20 世纪 80 年代时，Bakun 等人通过研究圣安德丝断裂层以及卡萨迪亚地震带的地质特点发现了某种特征地震的地震周期，从而发明了经典的地震预测方法，通过几次地震之间的时间间隔来预测地震的来临时间[1]。Bertrand Rouet‐Leduc 团队通过使用机器学习模型分析模拟地震实验室的实验室地震数据的过程中发现，使用机器学习模型预测实验室地震能带来相当精确的结果[2]。本文在 Bertrand Rouet‐Leduc 已有随机森林模型的基础上，首先对数据进行离散化处理，使用时间窗口工具，将每 1000 行数据抽象为一个数据点，得到了拥有三个特征值和一个目标值的训练集与测试集。此外，分别使用决策树模型，kNN 模型，朴素贝叶斯模型和神经网络模型对数据集进行分析和预测，对分别得到的不同的预测结果进行分析与评价。接着，通过添加决策树模型，kNN 模型，朴素贝叶斯模型以及神经网络模型构建集成学习模型并对特定的数据集进行预测。最终在特定的数据集上，通过建立多元分类器的集成学习模型得到了比随机森林更好的结果。在实验过程中，因为数据集的数量巨大，我也使用了 Spark 作为数据处理的工具，并且也因此得到了比普通编程软件快许多的数据处理速度。

关键词： 地震预测，实验室地震，机器学习，随机森林，集成学习

Time-Remaining Prediction of Laboratory Earthquakes

Abstract

Predicting the size, location and time-remaining of earthquakes has always been one of the most basic academic goals of geoscientists. Bakun et al. discovered the seismic period of a certain characteristic earthquake by studying the geological characteristics of the St. Anders fault zone and the Casadia seismic zone. Bertrand Rouet-Leduc team found that the use of machine learning models to predict laboratory seismic energy is quite accurate by using machine learning models to analyze laboratory seismic data from seismic laboratories. Based on Bertrand Rouet-Leduc's existing random forest model, data is discretized at first by this paper, using time window tool. Each 1000 rows of data is abstracted into one data point, and a training set and a test set with three eigenvalues and one target value are obtained. Also, decision tree model, kNN model, naive Bayesian model and neural network model are used to analyze and predict the dataset, respectively, and the different prediction results obtained are analyzed and evaluated. Finally an ensemble learning model was obtained by by adding decision tree model, kNN model, naive Bayesian model and neural network model altogether. Better results were obtained by establishing an ensemble learning model of the multivariate classifier. During the experiment, because of the massive data sets, Spark are also used as a data processing tool, and thus got much faster data processing speed than ordinary programming software.

Key Words： Earthquakes， Laboratory Earthquakes， Machine Learning，Random Forests， Ensemble Learning

目　录

1 引　言

能够预测地震的大小，地理位置和来临时间一直都是地质科学家最基本的目标之一。同时，预测地震可以避免许多人员的伤亡和财产上的损失。但是，很长一段时间以来，人类在地震预测这个问题上一直没有得到满意的答案。20 世纪 80 年代，Schwartz 等人在对圣安德烈斯断层以及卡萨迪亚俯冲带的研究中发现了一种具有特殊性质的地震——慢地震[3]。慢地震事实上是一种能量释放缓慢的地震，当地壳缓慢地移动交错时，能量会不断积累，当积累到一定程度上就会发生“黏滑失效”，于是地震就发生了。这种地震的特点之一就是这种地震具有相对于其他地震而言比较明显的周期性。所以在之后的几年，1895 年时，Bakun 等人在已有研究的基础上通过对上一次发生的慢滑地震的时间以及之前的地貌进行研究，得到结果：下一次地震有可能会在 1988 年到 1993 年这之间发生。结果是经过了十几年的时间，在 2004 年这一场地震才真正发生。所以误差还是相当大的。为了能够在地壳的活动中对“慢滑”以及“黏滑失效”这两个概念能够有更好的研究[4]，Johnson 等人沿着前人的思路在实验室中建立了可以模拟地壳运动的仪器。在通过实验仪器对地壳进行模拟的过程中，Johnson 等人通过精密的测量仪器记录下来了这个过程中出现的种种特征，得到了很多宝贵的数据[5][6][7][8]。在这之后，随着近些年来数据科学领域的蓬勃发展，Bertrand Rouet-Leduc 团队通过使用机器学习模型随机森林模型对这些实验室地震的数据进行了研究。通过随机森林模型，Bertrand Rouet-Leduc 团队得到了相当好的预测效果。并且在 Bertrand Rouet-Leduc 团队之后的研究中发现，这一方法在预测雪崩，山体滑坡，机器零件发生故障这些方面表现的也很出色。Bertrand Rouet-Leduc 团队使用的机器学习模型随机森林模型是一种集成学习模型，通过反复生成决策树来完成预测。我希望通过建立复合多元的集成学习模型来观察是否有可能得到更好的预测效果。在具体建立模型的过程中，我首先建立了决策树模型，最近邻模型，朴素贝叶斯模型，神经网络模型。并通过这四个模型对特定的数据集进行预测，观察预测效果并对预测模型做出评价。之后通过预测的结果将几个模型进行组合，建立集成学习模型。通过建立随机森林模型对特定的数据集进行预测得到预测效果，并且成为最终集成学习模型预测效果的判断标准。通过建立的集成模型进行预测，将得到的结果与随机森林结果进行比对，然后将结果进行分析与讨论。

2 文献综述

2.1 经典预测地震的方法

经典地震预测方法来源于 20 世纪 80 年代对圣安德烈斯断层的研究。圣安德烈斯断层的形成是由于两个板块交界处发生了相对的移动，当岩石的压力不断聚集，能量最终释放时，地震便发生了。在这个过程中，板块之间不断地运动，不断积累能量，释放能量导致了这种特征性地震是表现出了一定的周期性的特点的。圣安德烈斯断层带的古地震学数据导致了特征地震模型的形成。在对圣安德烈斯地震最近 10000 年的沉积岩的研究中，科学家们发现卡萨迪亚俯冲带已经积累了巨大的能量[2]，极有可能在短期内形成发生下一次地震的条件。在这个背景下 Parkfield 做出了著名的 Parkfield 猜想。Parkfield 猜想的重要基础便是特征性地震是周期性发生的，虽然周期并不是那么确定。在 1857 至 1986 年之间发生的所有相似的特征性地震的数据表明，特征性地震的周期大约为 18.8 年至 25 年之间，也就是说在 1988 年至 1993 年之间极有可能会在这里发生一次地震。虽然这次地震的实际发生时间要比预估时间晚，在 2004 年，但是这依旧为科学家提供了研究地震的宝贵思路。

2.2 用机器学习算法研究实验室地震

2.2.1 用机器学习算法对实验室地震进行预测

机器学习提供了各种算法，适用于对输入数据与输出数据之间的关系进行建模。但是在具体的运行中，通过随机森林进行建模的效果是最好的。一个经过训练的随机森林门票行能够通过时间窗口中的特征预测下一次地震来临之前的剩余时间。为了建立随机森林模型，首先通过时间窗口计算实验数据的统计特征。接下来将整个数据集划分为训练集和测试集。其中训练集用来生成随机森林模型，测试集用来检测实验结果。随机森林的最终结果是通过生成一组决策树，每一棵决策树都对一组数据做出一次预测，最终将这一组数据取平均值或众数作为最终的结果。

一个单一的决策树可以通过以下方式生成。为了能够预测距离下一次地震还有多少时间，由一个 root 节点出发并朝着 leave 节点来进行工作。任意一个决策树的内部节点都代表了一次选择。举例来说，决策树 root 节点可以是“在这一时间窗口中，方差值是否要大于 0.5”？如果答案是肯定的话，就

继续走向左边的分支，否则就走向右边的分支。这一系列决定会一直进行直到走到某一个 leaf 节点。每一个 leaf 节点都包含一个可能的预测值。所以每一棵决策树都代表了一个从输入到输出的地图[2]。

Bertrand Rouet-Leduc 使用随机森林的 Scikit-learn 来进行随机森林的运算。他们团队首先随机选择了时间窗口中统计值相关的一百个特征值（如均值,）来训练随机森林模型并做出预测，他们设置每个随机森林建树数量为 1000 课来完成对数据的分析。

他们使用的机器学习模型是随机森林，也是集成学习模型的一种，通过建立多个决策树分类器组合完成预测的任务。

2.2.2 用机器学习算法发现地震信号

地震破裂的这一现象与自然界的方方面面都有很深刻的联系。但是长期以来出于仪器或计算方法的不足，人类一直都未能探明这种关系。Bertrand Rouet-Leduc 使用机器学习算法随地震信号进行了分析。发现可以使用机器学习算法来探明地震信号中关于“摩擦失效”的特定模式。在板块之间缓慢运动来客服阻力摩擦，并且当这种力量忽然释放时，构造地震就发生了。事实上，推断板块的摩擦状态，以及现在在周期地震的哪个状态是极具有挑战性的。地震发生时的破裂速度，地震震级都可以用来做计算的基本参数。在实验过程中 Bertrand Rouet-Leduc 团队使用了梯度增强树的机器学习算法[9]。他们通过倒出数据时间窗口的统计特征，并应用了梯度增强树的算法，准确地从实验数据的瞬时特征中读出了瞬时剪切应力和摩擦状态。他们发现，任意数据段的统计特征能很好的分辨地壳所处的摩擦状态。他们将与地震动力学相联系的特征变量概括为了简单的状态方程。结果表明，在实验室地震模拟器中，地震并不是随机发生的，而是遵循特定的模式。他们还发现，在地震数据中有一个关键的信号能够对地震的来临做出预测，但是这一信号一直以来都被人类科学家当作“噪声”，这也说明机器学习模型在一定程度上可以克服人类的偏见，发现自然规律。

2.2.3 用机器学习算法对真实地震数据进行分析

从普通地震的慢滑到地震断层的蠕变，板块之间的滑动是以不同的方式呈现的。慢滑以及相关的震动模式在许多俯冲带上都是具有共性的。在一个很明显的例子中，卡萨迪亚地震带在慢滑的过程中传递出了离散的震动信号，这个振动信号能够精确地告诉板块之间的摩擦状态，可以预测地震的来临。

Bertrand Rouet-Leduc 团队在基于先前开发的机器学习算法的基础上，分析了来自温哥华的大量的原始的地震数据[5]，将这一震动信号与背景地震噪声信号分离。发现通过这种方式可以实时得接收来自于板块之间的摩擦状态，说明这种方法确实在确定地震来临方面是有效的。在这一模型的运用过程中，他们依旧使用了时间窗口的方法来对地震数据集进行了分析，但是与之前实验不同的是他们在这一实验中使用了大量来自于 GPS 的数据，并且在实验过程中，他们建立了许多不同大小的时间窗口的模型来进行比较。在这一次实验中他们不仅仅使用了随机森林算法，也使用了一些“黑盒”方法，如神经网络算法。最终得到的结果发现使用这种方法的确是可以获得板块之间实时的摩擦状态的。

3 背景知识

3.1 实验室地震装置

为了能够对在特征地震中产生的一些比较特殊的现象（如“慢滑”和“黏滑失效”）有更细致的研究，其中 DM Saffer 团队在研究过程中通过在实验室中建立装置完成一系列实验来对这两个现象进行研究，并且记录下来信息和数据。DM Saffer 团队通过在实验室中进行建模，得到了现实世界中地震发生时的一种抽象，进而发现了有关于“黏滑失效”的物理学及其动力学的一些关键性的见解[10]。

在实验中，实验使用双故障配置，由三个材质块以及夹在其中的两个凹槽组成。剪切轴上的位移通过直流电位仪传感器（DCDT）来进行测量。实验中使用刚或金属钛来制备固定块，钢或丙烯酸来制备剪切块。中间主要使用 Min-U-Sil 40 粉末二氧化硅（US Silica Co.）来模拟颗粒状断层泥。该产物为 99.5％SiO_2，在室温下 100%的相对湿度下剪切[10]。

将样品放入试验机后，施加恒定的法向的应力并使用力反馈伺服器来控制力保持恒定。在剪切开始之前，允许样品进行压实并适应颗粒重排。通过使用反馈伺服器控制中心受力块以固定的速度进行位移来诱导剪切。在实验过程中记录所有产生的声学信号，将会作为之后数据分析数据的来源。声学信号是以极高的频率收集的，所以能够带来相比较而言更多的信息来进行分析。

3.2 决策树模型

决策树（或分类树）是已知目前最直观也是最常使用的分类模型之一。这是一种监督学习模型[11]，建立模型的目的在于当拥有一组数据特征时可以通过生成的模型可以进行一系列判断，最终达到进行预测的目的。决策树模型的原理是，通过关键的参数如信息熵（Entropy）和 Gini 系数[11]，在一个关键的过程成为信息[12]增益的过程之中生成一系列判断规则，然后生成最终的模型。其中，信息熵的定义为：

$$H(X_m) = -\sum_k p_{mk} \log(p_{mk}) \quad (3.1)$$

Gini 系数的定义为：

$$H(X_m) = \sum_k p_{mk}(1 - p_{mk}) \quad (3.2)$$

但是这一模型也具有一些缺点，它在建树的过程中有可能会将树建的过于复杂以致于并不能很好地迭代数据[13]，而且在使用很复杂的数据集时也很有可能会产生过拟合的问题。也就是说，这一算法在训练集上会有很好的表现，但是在预测集上就没有办法达到这样了。

决策树模型已知比较常用的算法有 ID3、C4.5 和 CART 算法这三种。其中 ID3 算法是由 Ross Quinlan 在 1986 年提出的。这个算法是通过贪婪算法，希望得到一个信息增益最大的节点，并不断迭代生成一颗完整的树。在建树完成后再通过树类型的运算来进行修饰加强树的表达能力。C4.5 算法是 ID3 算法的继承者，C4.5 算法通过动态的添加属性，移除了特征必须是明确的这一要求。C4.5 将建好的树转化为一组“如果那么”规则。这些规则会通过评估确定应用的节点和位置。CART 是一种与 C4.5 很相近的算法，但是不同之处在于 CART 算法支持数值类型的目标变量并且不对决策规则进行计算。CART 算法通过运用特征建立二叉树并且让每个节点的信息增量值达到最大。通常构造平衡二叉树的时间复杂度为：$O(n_{samples}n_{features}\log(n_{samples}))$ 查询时间为：$O(\log(n_{samples}))$。

我在使用决策树模型算法时，主要使用的是 Scikit-Learn 中的决策树生成算法[14]，也就是 CART 算法，来对我的数据进行分析。

3.3 kNN 模型

最近邻算法可以同时为监督性学习和非监督性学习提供帮助。非监督性的最近邻算法是许多其他学习方法的基础，比如流形学习和谱聚类。基于最近邻的监督性学习有两种使用方式：通过离散的标签进行分类和通过连续的标签来进行回归。

这一算法背后的原理是通过计算距离，来找出在样本中和已知点最近的几个点，然后通过这些点来对已知点进行预测[15]。样本点数量的选取可以是用户自定义的，也可以通过样本点附件点的密度来确定。距离的定义一般为欧几里得距离。这一方法普遍被认为是非泛化的学习方式。因为他们在计算过程中会使用所有的数据。除了原理比较简单，这一方法也能够成功的使用在很大的数据集上。在分类边界很不规则时，这作为一种无参的方法也能得到很好的效果。

我将使用 Scikit-Learn 中的最近邻算法来完成我的数学实验[14]。

Scikit-Learn 中主要提供两种分类方法：通过计算周围 k 个最近的点，或是根据固定的半径 r 计算每个训练点周围样本点的个数[15]。一般计算周围 k 个最近的点是最常用的技术。在这个过程中 k 的选择是非常依赖于数据的。但是总体来说更大的 k 可以减少噪声的影响，但是让分类边界变得不那么明

显。

当数据没有被均匀采样时，采用固定半径的分类方法的效果会更好。但是对于高维度的参数空间，这种方法就不会那么有效。

最近邻模型已知的比较常用的算法有暴力计算、K-D Tree、Ball Tree。通过快速计算来得到最近的几个样本点是机器学习最活跃的领域之一。最初使用暴力计算得到最近邻是通过搜索整个数据集，计算所点到点之间的距离。对于小的数据样本来说，这一种方法是很有效的。但是当样本的数量增加时，这一方法很快就变得没有实现的可能了。所以为了能够解决暴力算法效率低下的问题，发明了很多树形结构。这些结构尝试通过高效地编码样本之间的距离的信息来减少所需的距离运算的次数。最基本的想法是当点 A 距离点 B 很远并且 C 又距离点 B 很近时，我们就可以知道点 C 距离 A 也很远，并不需要精确的计算，这样可以减少很多的运算量。这是对于暴力算法的一个很大的改进。最早的使用这种方法来获取信息的方法是 K 维树，k-D 树是一种二叉树，并且预算效率很高，但是当数据的维度升高至 20 以上时时 KD 树的运算就会变得很慢。Ball Tree 是为了解决 KD 树在高维度时运算效率低下的问题被发明的。在 KD 树沿 Cartesian 轴进行分割时，Ball Tree 会将数据分成一系列嵌套超球。这使得 Ball Tree 的建造成本要比 KD 树高许多，但是在高维度的情形下 Ball Tree 建造模型会更加有效。Ball Tree 通过递归将数据划分到以 C 为中心半径为 r 的点中。通过三角不等式来计算所需要搜索的邻近点的个数。通过这样的规划从中心点到邻近点的上限和下限便可通过单一的距离计算得到。

3.4 朴素贝叶斯模型

朴素贝叶斯方法其实是一组基于贝叶斯定理和基本假设每一组特征之间相互无关的监督学习算法。贝叶斯定理可以用下面的关系叙述：变量 y 与相互不相关的自变量矩阵从 x_1 到 x_n 有： $P(y \mid x_1,\ldots,x_n)=\frac{P(y)P(x_1,\ldots x_n \mid y)}{P(x_1,\ldots,x_n)}$ 。所以我们可以使用基于概率统计的分类规则。基于概率的分类器的主要区别就在于他们使用的概率假设是不同的，所以如果想使用基于概率统计的分类模型，就一定要选好概率模型。这一种分类器在绝大多数情形下都可以预测的十分良好，虽然基于这一种分类器进行的假设是十分简单的[14]。但是这一种分类器的优势在于，这一种分类器在复杂的模型上的时间复杂度很低[16]。所以也是具有优势的一种分类器。

3.5 神经网络模型

多层感知器（MLP）是一种监督性学习算法，它的实现是通过接收数据集的训练学习方程：$f(\cdot): R^m \to R^o$。其中 m 是输入集的维数，o 是输出集的维数。当接收到一组特征值 $X = \{x_1, x_2, ..., x_m\}$ 以及模型的目标值 y 时，这一模型可以通过计算一组非线性方程来进行分类或回归问题的预测。这一算法与 Logistic 回归有一些不同。这里的输入与输出之间可以具有许多层级，这些层级可以是线性或者非线性层，被称为隐藏层[14]。这一模型可以计算非线性模型并且能够进行实时学习。但是随之而来的这一模型也需要调整更多的参数，而且也对特征缩放很敏感。这一模型能够使用的算法有随机梯度下降、Adam 或 L-BFGS 算法。Adam 与 SGD 相似都是随机优化器，但是 Adam 可以根据低阶矩的自适应估计自动调整更新参数的数量。L-BFGS 是一个代表函数的二阶偏导数的近似与海瑟矩阵的求解器。并且，它模仿海瑟矩阵的逆来完成参数的更新。

3.6 随机森林模型

随机森林是一种集成学习算法[17]，因为它的单个元素是决策树，所以随机森林也是一种监督学习模型[18]。这一模型的背后思想是基于概率模型，当一个分类器的分类效果并不是那么好的时候，通过增加分类器的个数，从众多分类器的结果中挑选平均数或者众数能够获得更好的结果[19]。这一种方法一般需要对不同的特征进行测试，也需要特征选择的过程，来达到这一算法的最佳效果。在 Bertrand Rouet-Leduc 的实验中，他们建立的随机森林每一个随机森林模型包含 1000 棵决策树。

3.7 集成学习模型

集成学习模型的目标就是希望通过一组多个分类器来增强预测的实用性和鲁棒性。随机森林就是集成学习的一种[20]。这一模型中可以使用不同的分类器也可以使用相同的分类器，来对预测目标进行预测。这一模型可以做分类预测也可以做回归预测。在对不同分类器的结果中，可以通过建立多个相互独立的分类器取众数[21]，或是可以通过建立多个相互独立的分类器然后对结果进行平均。一般来说估计效果会比单一的估计器效果要好，比如套代法和随机森林使用的都是这种方法。还有一种是增强法，估计器是按照顺序构建的，比如 AdaBoost 使用的便是这个方法。在集成学习的算法中，套袋法组

成了一类算法，这些算法通过在随机的数据子集上建立多个黑盒实例然后将它们组合在一起完成最后的预测。这些方法主要用来减少一个基础分类器的方差。原理就是通过在它构造的过程中加入随机量，并且将所有的结果组合在一起。在许多情形下，套袋方法都提供了一个在单一分类器的基础上提升模型效果的简单方法，因为这一方法可以有效的减少模型的过拟合问题，所以在面临过于复杂的问题时这一方法十分适用。在这一算法的分类中，当数据是来自于随机抽取的数据集时使用 Pasting 算法。Adaboost 算法的核心理念是基于一组弱分类器不断地对数据进行分析。所有的分类器按照一定的权重公式组合起来。在模型刚开始建立时，将所有的权重设为一样的平均值，之后在每一次的迭代中不断修改权重，然后再将算法重新应用在数据上直至达到最优[14]。

3.8 大数据计算引擎 Spark

Apache Spark 是一个包含许多工具包并且可以在一个计算机集群上进行并行计算的复合计算引擎[22]。Spark 是一种开源的软件，适用于所有想要做大数据科学的数学家或工程师。应用程序开发人员和数据科学家将 Spark 整合到他们的应用程序中，以便大规模快速查询，分析和转换数据。与 Spark 最常关联的任务包括跨大型数据集的 ETL 和 SQL 批处理作业，处理来自传感器，物联网或财务系统的流数据以及机器学习任务。Spark 可以使用多种不同的语言进行编程，并且可以从数据库中读入数据和流数据。Spark 的核心目标是希望建立一个拥有复合功能的计算平台，并且提供了多种 API 可供使用。各种工具包也提供了在上层更容易使用机器学习算法的可能。Spark 可以部署在多种存储平台上，比如 Azure Storage，Apache Hadoop。作为一种数据计算引擎，与 Apache Hadoop 不同的是，Apache Spark 将更多的关注点放在计算上，它的计算能力比 Apache Hadoop 的 MapReduce 计算架构更强。在谷歌发布 MapReduce 白皮书之后 Doug Cutting 和 Mike Cafarella 创造了 Hadoop。Apache Spark 在 2009 年作为加州大学伯克利分校 AMP 实验室的项目被发布。Spark 在 2013 年成为 Apache 软件基金会的孵化项目，并在 2014 年初推出，成为基金会的顶级项目之一，Spark 目前获得了众多个人以及企业的支持比如 IBM 和华为。Spark 项目的目标是保持 MapReduce 可扩展分布式，容错处理框架的优势，使其更高效更易于使用。Spark 对于使用迭代算法并行处理分布式数据很有用。

Spark 的运行速度可以比上一代 Hadoop 的计算引擎 MapReduce 快 10 至 100 倍，是一种很好用的数据计算引擎。Spark 的数据文件是被分布式存储

的，每当有数据集上传至计算机都会被分成小的数据块然后分布在各个数据节点上，这能够为 Spark 带来很高的容错性。Spark 运行计算时都是通过分布式计算，用户在平台上写的方程会被自动分发给不同的集群来进行运算。之前分布存储的数据会被运输至工作节点来进行运算，所以会带来很高的运算速度。

3.9 模型性能评估方法

模型的评估可以从很多不同的角度来进行，比如准确率，方差。使用的方法也有很多。但是在这里的研究中，我将采用平均绝对误差 MAE（Mean Absolute Error）来对模型进行评估。在统计学中，MAE 是对两组连续变量之间不同程度的测量。如果有 n 个数据点，每一个数据点都代表了一组数据 (x_i, y_i)，那么 MAE 值可以通过式子 $\mathrm{MAE} = \frac{\sum_{i=1}^{n} |y_i - x_i|}{n} = \frac{\sum_{i=1}^{n} |e_i|}{n}$. 给出。它在我的实验中可以代表预测值与实际值之间的差距。当算出来的值越大时，预测值与实际值值之间的差距就越大，反过来当计算出来的值越小时，预测值与实际值之间的差距也越小。平均绝对误差是将预测与最终结果进行比较的多种方法之一。一些可以替代的方法也有平均绝对标准误差和均方误差。

4 实验方法

4.1 原始数据

这些数据来自于洛斯阿拉莫斯国家实验室[4]。在双轴直接剪切几何形状的岩石上进行的双轴加载实验，一个典型的实验室地震模型。在承受恒定的法向载荷和规定的剪切速度的同时，同时剪切两个断层泥层。实验室故障在重复的粘滞和滑移循环中失效，这意味着模拟构造断层的加载和失效循环。虽然实验比地球上的故障简单得多，但它具有许多物理特征。洛斯阿拉莫斯的初步工作表明，在准周期性实验室地震周期的情况下，可以从连续地震数据预测实验室地震。在这个数据集中非周期性地震更多。

数据为.csv 文件，大小为 8.9 Gb，一共两列：其中一列 acoustic_data 是作为输入数据用来预测距离下一次地震剩余的时间 time_to_failure。其中前一列数据是输入数据，后一列数据是目标变量。

表 4-1：数据样例

acoustic_data	time_to_failure
12	1.4691
6	1.4691
8	1.4691
5	1.4691
8	1.4691
8	1.4691
9	1.4691
7	1.4691
-5	1.4691
3	1.4691
5	1.4691
2	1.4691
2	1.4691
3	1.4691
-1	1.4691
5	1.4691
6	1.4691
4	1.4691

3	1.4691
5	1.4691

4.2 数据预处理

为了对数据有一个大致的了解，我首先使用 Spark 对实验数据进行了统计。

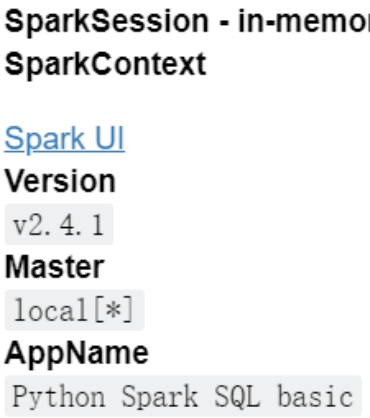

图 4-1 Spark 运行结果

统计数据为数据一共有 600000000000000 行，一共运行时间为 1067 秒。因为数据的数量太大所以我不可能将所有数据运行。

4.3 数据离散化

为了增加运行的可能首先对数据进行截取，首先在训练集中计算得到所有的目标值，选取目标值 0 秒至 1.4 秒，寻找测试集中有相同目标值的位置，在训练集中也截取数据 5500000 行，并找出目标值相同的测试集 5500000 行，准备好训练集和预测集。整个训练集中的地震周期最大值为大约 14 秒，截取其中 1.4 秒作为目标，因为当对所有数据进行处理时平均每行的数据处理时间为大约 500 秒。因为分类算法是无法接受小数作为目标，所以我的解决方法是通过小数乘 10 后取整后作为目标值。所以一共有 15 个目标分类。当乘以 100 然后取整，会得到 150 个目标分类，这样的话训练所有的机器学习模型的过程就会变得极度复杂，模型的训练时间和复杂度都会呈非线性增长。并且就实际情形来说，在这个问题上用很高的时间和计算机运行的代价获得一点点精度的提升也是并不划算的。所以一共使用 15 个目标值作为分类。在确定时间窗口的大小时也是一样的。我选取每 1000 行作为一个时间窗口。当

选择 100 行时间窗口时，5000000 数据就会一共有 50000 个时间窗口，这样的话可以在更少的信息中获取更多的数据，但是同时增大的也有运算量，生成的决策树模型和其他分类模型就会变得很复杂。同时，虽然采用更小的时间窗口可能会获得更多的信息，但是并不是所有的信息都是实验中需要的。如果采用过大的时间窗口的话，虽然可以减小运算量[24]，但是时间窗口里包含的特征信息也会被相邻的目标变量相互抵消，所以我使用 1000 作为时间窗口的大小。在选取时间窗口的特征值时，选取均值，方差和极差作为时间窗口的特征值。在实验中 Bertrand Rouet-Leduc 一共选取了大约 100 个特征值进行实验，然后通过特征选择的方式将特征减少，减少为 50 个关键特征[4]。出于运算成本的考虑，选取更多的特征可能会让模型的训练变得异常困难，在这种背景下至选用了三个特征值，也是统计中最基本的三个特征值。然后通过运算得到了一个拥有 5399 行数据的训练集和拥有同样行数的测试集。最终数据预处理的时间为大约 12 小时。

表 4-2：时间窗口训练集特征值

mean	var	diff
5.137	22.41823	57
5.35	34.9935	48
5.015	712.1428	202
5.364	295.5615	123
4.925	105.2494	71
5.138	15.72496	25
5.261	13.49288	27
5.346	14.92428	31
5.143	32.82655	38
5.141	8.347119	18
4.921	7.984759	20
4.842	16.36304	30
4.767	16.26071	27
4.862	59.95696	60
5.043	39.48515	51
4.984	11.03774	21
5.18	15.2876	40
4.814	62.2174	50
5.056	9.530864	23
4.705	7.679975	17

表 4-3：时间窗口训练集目标值

target
14
14
14
14
14
14
14
14
14
14
14
14
14
14
14
14
14
14
14
14
14

表 4-4：时间窗口测试集特征值

mean	var	diff
4.559	43.56252	55
4.69	14.7039	27
4.615	11.62878	22
4.481	8.447639	16
4.778	30.19672	62
4.69	227.9339	99
4.271	13.53956	26
4.307	8.844751	21
4.8	11.49	22

4.582	190.8133	117
4.606	47.17476	51
4.665	13.41078	27
4.595	9.594975	21
4.653	11.14859	25
4.758	14.23344	25
4.317	13.54051	23
4.546	28.66988	54
4.622	23.21312	31
4.598	9.400396	18
4.661	9.094079	20

表 4-5：时间窗口测试集目标值

target
14
14
14
14
14
14
14
14
14
14
14
14
14
14
14
14
14
14
14
14
14
14

4.4 决策树模型的构建

在实验中使用 Scikit-Learn 作为构建决策树的工具。Scikit-Learn 使用 CART 算法构建决策树模型。因为这是对单个模型的预测结果进行评价，所以并不那么考虑构建模型的时间成本，希望决策树模型能够得到尽量多的信息，所以将 5399 行数据输入模型进行训练，在训练好模型后对测试集进行预测，将结果输出为.csv 文件并计算 MAE 值对结果进行评价。因为对于同一组训练集得到的决策树有可能会是不同的。所以在运行中一共生成 10 组决策树来对测试集做出预测并取平均值作为决策树模型的结果。在生成决策树的过程中也需要在决策树输入相应的参数，比如建树的深度，所有目标值的个数，在实际创建时，保持这些值为默认值[25][26][27]。

4.5 kNN 模型的构建

kNN 模型是一种基于距离的分类模型，当使用的距离的性质不同时得到的 kNN 模型的结果也是不同的。所以在这一模型建立的过程中也不只需要数据作为参数的输入，也需要选择距离的类型，以及目标值的个数。在这个问题中，使用欧式距离来做训练。但是因为 kNN 模型中对于同一组数据集训练的 kNN 模型是一样的，所以并不需要取多次平均来验证结果。但是在实际实验的过程中，为了防止模型出现过拟合或者是别的其他问题，我们对同一模型也对训练集进行测试，并将两组结果进行比对。出于希望在一个模型输入尽量多的信息的考虑，将所有的训练集的数据输入到模型中进行训练。

4.6 朴素贝叶斯模型的构建

这一模型是一种对简单问题分类可以做到相当出色的分类器，但是在复杂关系的处理上，这个模型可能便不会有那么好的效果。在 Scikit-Learn 中可以使用的工具包，有 Gaussian Naive Bayes、Complement Naive Bayes 和 Bernoulli Naive Bayes。但是在实际使用的过程中只有 Gaussian Naive Bayes 模型是对本问题适用的。在这个问题上适合的概率模型就是高斯概率分布。这一模型与 kNN 模型相同，对于相同的训练集，测试得到的结果是相同的，所以也并不需要重复构建模型，而是选择输入尽量多的信息来对模型进行训练查看训练的结果。同时，也对训练集进行预测，通过两个结果的相互比对来看这一模型是否在运行过程中产生了过拟合。

4.7 神经网络模型的构建

神经网络模型是一种黑盒模型，在训练的过程中人类并不能了解它背后的发现的相关问题的哲学和知识。并且神经网络还有的特点是运算可能会非常复杂，在输入层与输出层之间可能会有许多隐藏层，这些计算会非常占用计算机的内存与时间。所以这一算法模型并不太适用于大的数据集。如果想要使用神经网络模型应该使用如 TensorFlow 这样的框架来进行计算。但是因为实验中的数据集是已经被简化过的，所以使用 Scikit-Learn 中的神经网络模型也可以对这些数据集进行分析。在实际的运用中，可以选择隐藏的层数以及输出个数的数量，来进行分类。在这一模型的训练中，我们依旧同时对测试集的训练集做出预测，通过训练集和测试集之间的对比，来观察这一种算法模型的适应性。

5 集成学习模型的构建

5.1 随机森林模型的构建

在实验进行的过程中，我选用与Bertrand Rouet-Leduc团队相同的方式，通过 Scikit-Learn 的随机森林算法来对数据集进行分析和预测。建造这一模型在实验中主要是为了首先通过建造不同数量的森林来观察森林模型的预测效果，并且森林模型的预测结果也可以为相对而言本实验建造的集成学习模型进行参考。在森林模型的训练过程中，不仅仅需要对森林的参数进行调整，也可以对树的参数进行调整。为了能够统一的观察到不同的算法模型的效果，在这类的森林模型也使用和决策树模型时同样的建树的参数。

5.2 集成学习模型的构建

目前的集成学习模型的核心思想是通过结合一组分类器，让这一组分类器对同一个问题做出答案，再将这些答案进行组合和研究，来获得一个更好的预测效果。其中森林模型是通过结合一组决策树模型的结果来丰富预测效果的。也可以使用通过结合不同的分类器来对实验做出比较好的预测效果。在已经进行过的实验以及他们的预测效果，我们可以发现在四种模型中决策树的预测效果是最好的，因为决策树的决策结构可以在同样的特征结构的基础上建立更复杂的联系，但是相应的结果是建立的模型可能会很复杂，在当在实验中加入更多的特征值时，模型可能会变得异常复杂，消耗的时间也会更长。kNN 模型在这个实验中的结果是仅次于决策树模型的，并且 kNN 模型相较于决策树模型来说，创建的模型的结果更加简单，在数据训练集较小时这一个方法能够很好的逼近决策树的预测效果，但是当实验中使用的训练集的数量级增大时，这一算法的时间复杂度就会变得很高，并且当加入更多的特征进入模型时，这一算法模型也不一定能够良好的描述这些特征与目标值之间的联系。NB 模型作为一种概率模型，时间复杂度很小，是一种很简单的模型，不管数据量与特征值的数量，都能够很容易进行训练和预测，但是模型过于简单的结果便是，NB 模型并不是很擅长描述复杂的特征关系以及和目标值之间的关系，所以使用 NB 模型来预测的结果也是最差的。神经网络模型是一种黑盒模型，在训练模型的过程中以及之后训练出来的结果，人类只能对这一结果进行使用，却并不能够进行解释。这一模型可以在输入层与输出层之间加入许多隐藏层，来描述非线性的结构。同样的，当输入更

多的数据进入训练集或者添加更多的特征时，这一模型的时间复杂度就会上升，对运算的要求也会变得越来越高。

在构建集成学习模型的过程中，首先需要抽样，因为训练的模型增多了，所以模型的训练成本也会增大，通过减少数据的训练集的大小就可以使用相同的数据集训练更多地模型。所以首先确定总体训练集的 10%进行取样作为一个分类器的训练器，也就是 540 个样本点。这 540 个样本点通过抽取随机数从总的训练集中抽出，然后对模型进行训练。每一种方法训练出 100 个分类器，然后这些分类器在各个相互独立的分类器之间取平均数作为最终分类的结果。这四个方法每一个方法做同样的操作然后得到最终的结果。

6 实验结果分析与讨论

6.1 决策树模型构建结果

首先通过训练得到了决策树模型，生成好的决策树模型可以通过文字的方式写入.dot 文件中进行保存。

其中部分结果如下：

digraph Tree {

node [shape = box, style = "filled, rounded", color = "black", fontname = helvetica] ;

edge [fontname = helvetica] ;

0 [label = "mean < = 4.812\ngini = 0.931\nsamples = 5400\nvalue = [128, 385, 385, 386, 385, 385, 385, 385, 385, 385\n385, 385, 385, 385, 266]", fillcolor = "#7be53900"] ;

1 [label = "var < = 8.269\ngini = 0.927\nsamples = 4199\nvalue = [128, 365, 372, 324, 314, 363, 346, 344, 318, 311\n305, 289, 138, 150, 132]", fillcolor = "#c0e53900"] ;

0 -> 1 [labeldistance = 2.5, labelangle = 45, headlabel = "True"] ;

2 [label = "mean < = 4.328\ngini = 0.887\nsamples = 868\nvalue = [84, 230, 31, 36, 60, 54, 39, 54, 51, 53, 46, 48\n23, 26, 33]", fillcolor = "#e5c5392f"] ;

1 -> 2 ;

3 [label = "var < = 7.493\ngini = 0.787\nsamples = 280\nvalue = [56, 111, 16, 9, 14, 11, 11, 12, 8, 13, 8, 6, 0\n2, 3]", fillcolor = "#e5c5393f"] ;

2 -> 3 ;

4 [label = "mean < = 3.865\ngini = 0.674\nsamples = 147\nvalue = [40, 73, 5, 3, 4, 2, 4, 2, 3, 5, 2, 1, 0, 1\n2]", fillcolor = "#e5c5394f"] ;

3 -> 4 ;

5 [label = "var < = 7.06\ngini = 0.486\nsamples = 12\nvalue = [8, 3, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0\n0]", fillcolor = "#e581398e"] ;

4 -> 5 ;

6 [label = "mean < = 3.694\ngini = 0.444\nsamples = 3\nvalue = [1, 2, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0\n0]", fillcolor = "#e5c5397f"] ;

5 -> 6 ;

7 [label = "gini = 0.0\nsamples = 1\nvalue = [1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0\n0]", fillcolor = "#e58139ff"] ;

6 -> 7 ;

8 [label = "gini = 0.0\nsamples = 2\nvalue = [0, 2, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0\n0]", fillcolor = "#e5c539ff"] ;

6 -> 8 ;

9 [label = "mean < = 3.731\ngini = 0.37\nsamples = 9\nvalue = [7, 1, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0\n0]", fillcolor = "#e58139bf"] ;

5 -> 9 ;

10 [label = "range < = 17.5\ngini = 0.5\nsamples = 2\nvalue = [1, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0\n0]", fillcolor = "#e5813900"] ;

并且可以通过.dot 文件生成决策树图形，让决策树有更直观的表现。

可视化结果如下：

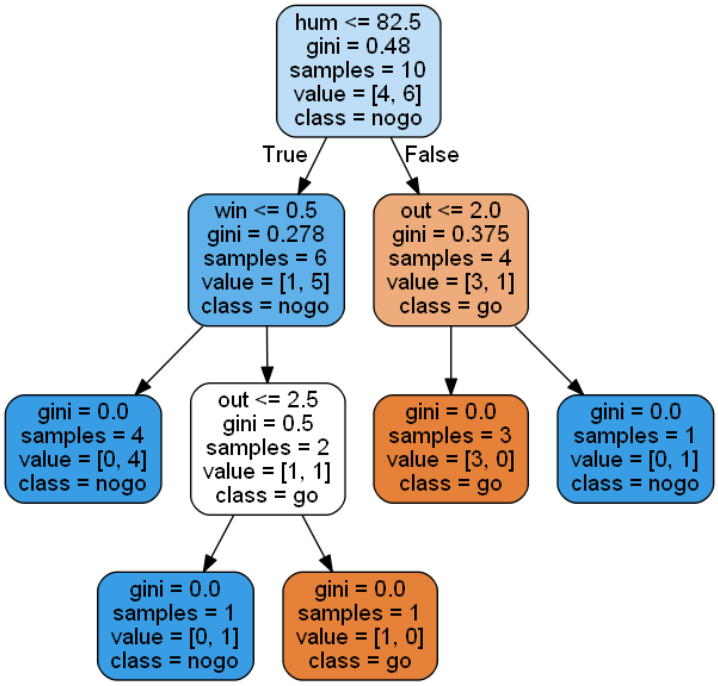

图 5-1：部分决策树图形

上图代表一颗小的决策树，图中包含的元素有矩形框和箭头。其中每一个矩形框代表一次决策，箭头代表决策的方向。最下层没有箭头指出的矩形框代表了叶节点，叶节点包含预测的结果，当走到叶节点时一次预测就结束了。每当输入一组特征数据时，根据矩形框中的文字进行的提示来进行判断，当矩形框中的内容判断为真时，走向矩形框左下的矩形框进行判断，当判断结果为否时走向右下角的矩形框，直至得到结果。

得到决策树对测试集的预测结果如下：

表 6-1：决策树 1 的预测值

预测值	实际值	误差	误差绝对值
4.00E+00	14	-1.00E+01	10
7.00E+00	14	-7.00E+00	7
1.10E+01	14	-3.00E+00	3
3.00E+00	14	-1.10E+01	11
3.00E+00	14	-1.10E+01	11
1.00E+01	14	-4.00E+00	4
6.00E+00	14	-8.00E+00	8
2.00E+00	14	-1.20E+01	12
1.20E+01	14	-2.00E+00	2
4.00E+00	14	-1.00E+01	10

其中第一列为预测值，第二列为实际值，第三列和第四列为误差。

一共生成 10 课决策树得到的结果如下：

表 6-2：所有决策树的预测结果

决策树	预测结果
决策树 1	4.139841
决策树 2	4.161326
决策树 3	4.151139
决策树 4	4.15188
决策树 5	4.148916
决策树 6	4.159659
决策树 7	4.153917
决策树 8	4.169661
决策树 9	4.161882
决策树 10	4.172995

最终得到的决策树预测结果的误差为：0.4157 秒。在之后的随机森林的预测中可以发现，决策树的预测与随机森林预测的结果是相近的，可能是因为使用不同的决策树取平均的过程与随机森林的建立过程是相近的。所以能够做到最终的结果也是相近的。在建立决策树的过程中，建立的决策树每个决策

树大致需要 8000 次判断，所以建立的决策树还是相对来说比较复杂的。

6.2 kNN 模型的构建结果

kNN 模型建立的结果如下，对测试集的预测结果为：

表 6-3：kNN 模型对测试集的预测结果

预测值	实际值	误差	误差绝对值
1.20E+01	14	-2.00E+00	2
9.00E+00	14	-5.00E+00	5
3.00E+00	14	-1.10E+01	11
3.00E+00	14	-1.10E+01	11
4.00E+00	14	-1.00E+01	10
3.00E+00	14	-1.10E+01	11
2.00E+00	14	-1.20E+01	12
5.00E+00	14	-9.00E+00	9
8.00E+00	14	-6.00E+00	6
3.00E+00	14	-1.10E+01	11
2.00E+00	14	-1.20E+01	12
1.20E+01	14	-2.00E+00	2
1.00E+01	14	-4.00E+00	4
4.00E+00	14	-1.00E+01	10
9.00E+00	14	-5.00E+00	5
5.00E+00	14	-9.00E+00	9
3.00E+00	14	-1.10E+01	11
4.00E+00	14	-1.00E+01	10
3.00E+00	14	-1.10E+01	11
6.00E+00	14	-8.00E+00	8

其中第一列为预测结果，第二列为实际值，第三列为误差，第四列为误差绝对值。所以测得预测的结果的误差值为 0.4212 秒。

对训练集的测试结果如下：

表 6-4：kNN 模型对训练集的预测结果

预测值	实际值	误差	误差绝对值

1.30E+01	14	-1.00E+00	1
2.00E+00	14	-1.20E+01	12
3.00E+00	14	-1.10E+01	11
9.00E+00	14	-5.00E+00	5
6.00E+00	14	-8.00E+00	8
1.40E+01	14	0.00E+00	0
1.20E+01	14	-2.00E+00	2
1.10E+01	14	-3.00E+00	3
2.00E+00	14	-1.20E+01	12
1.30E+01	14	-1.00E+00	1
1.40E+01	14	0.00E+00	0
8.00E+00	14	-6.00E+00	6
3.00E+00	14	-1.10E+01	11
9.00E+00	14	-5.00E+00	5
1.20E+01	14	-2.00E+00	2
1.20E+01	14	-2.00E+00	2
1.30E+01	14	-1.00E+00	1
1.10E+01	14	-3.00E+00	3
5.00E+00	14	-9.00E+00	9
5.00E+00	14	-9.00E+00	9

表格中第一列为预测值，第二列为实际值，第三列第四列为误差，所以最终测得这一模型对训练集本身的预测误差为：0.34688 秒。首先就这一模型本身的预测效果与决策树模型的对比来说，这一模型是稍逊于决策树模型的。因为决策树模型可以从相同的输入特征中猜测更复杂的关系，所以当单一的决策树与单一的 kNN 模型进行对比时决策树可能会更好一点。从模型对训练集和测试集的结果对比来看。训练集要比测试集的结果好很多，这也就说明这一模型在训练过程中很可能产生了过拟合状态，也就是说这一模型对训练集有点过于适应了，当面对一个新的测试集时，因为两个测试集各自拥有的特征是有区别的，所以对测试集的预测效果并不是很好。

6.3 朴素贝叶斯模型的构建结果

朴素贝叶斯模型的构建结果为：

表 6-5：NB 模型对测试集的预测结果

预测值	实际值	误差	误差绝对值
4.00E+00	14	-1.00E+01	10
1.00E+00	14	-1.30E+01	13
1.00E+00	14	-1.30E+01	13
1.00E+00	14	-1.30E+01	13
4.00E+00	14	-1.00E+01	10
6.00E+00	14	-8.00E+00	8
1.00E+00	14	-1.30E+01	13
1.00E+00	14	-1.30E+01	13
1.00E+00	14	-1.30E+01	13
6.00E+00	14	-8.00E+00	8
4.00E+00	14	-1.00E+01	10
1.00E+00	14	-1.30E+01	13
1.00E+00	14	-1.30E+01	13
1.00E+00	14	-1.30E+01	13
1.00E+00	14	-1.30E+01	13
1.00E+00	14	-1.30E+01	13
4.00E+00	14	-1.00E+01	10
1.00E+00	14	-1.30E+01	13
1.00E+00	14	-1.30E+01	13
1.00E+00	14	-1.30E+01	13

所以根据计算结果得到 NB 模型对预测误差值为 0.5502 秒。

NB 模型对训练集的预测结果如下：

表 6-6：NB 模型对训练集的预测结果

预测值	实际值	误差	误差绝对值
4.00E+00	14	1.00E+01	10
4.00E+00	14	-1.00E+01	10
1.00E+01	14	-4.00E+00	4
6.00E+00	14	-8.00E+00	8
7.00E+00	14	-7.00E+00	7
1.00E+00	14	-1.30E+01	13
1.00E+00	14	-1.30E+01	13
1.00E+00	14	-1.30E+01	13
1.00E+00	14	-1.30E+01	13
1.00E+00	14	-1.30E+01	13

1.00E+00	14	-1.30E+01	13
1.00E+00	14	-1.30E+01	13
1.00E+00	14	-1.30E+01	13
4.00E+00	14	-1.00E+01	10
4.00E+00	14	-1.00E+01	10
1.00E+00	14	-1.30E+01	13
1.00E+00	14	-1.30E+01	13
4.00E+00	14	-1.00E+01	10
1.00E+00	14	-1.30E+01	13
1.00E+00	14	-1.30E+01	13

所以 NB 模型最终计算得到误差值为 0.5618 秒。所以同它自己在训练集和测试集的表现上来说，这一模型并没有产生过拟合状态，在两个数据集上计算得到的是相似的。但是从 NB 模型与其他几个模型如决策树模型和 kNN 模型的对比来说，这一模型的预测效果是最不好的，可能是因为这一模型是所有模型里最不适合研究相对复杂的关系的，剩下几个模型更擅长处理这一种关系。在已有的三个特征中，kNN 模型和决策树模型更擅长描述三个特征值与目标值之间的关系。距离模型可能比概率模型更适合用来做关于这方面的研究。

6.4 神经网络模型的构建结果

出于对同一数据集不同参数的调整，也会为预测带来不同的结果。在这一模型的构建过程中是有许多参数都可以进行调整的，但是出于对运算量的考虑，所以在实验中并未使用太多的隐藏层。并且设置输出个数为 15。然后对模型进行训练并预测结果如下：

表 6-7：神经网络模型对测试集的预测结果

预测值	实际值	误差	误差绝对值
3.00E+00	14	-1.10E+01	11
1.10E+01	14	-3.00E+00	3
1.10E+01	14	-3.00E+00	3
9.00E+00	14	-5.00E+00	5
1.00E+00	14	-1.30E+01	13
3.00E+00	14	-1.10E+01	11
1.10E+01	14	-3.00E+00	3

1.00E+00	14	-1.30E+01	13
1.10E+01	14	-3.00E+00	3
3.00E+00	14	-1.10E+01	11
3.00E+00	14	-1.10E+01	11
1.10E+01	14	-3.00E+00	3
1.10E+01	14	-3.00E+00	3
1.10E+01	14	-3.00E+00	3
6.00E+00	14	-8.00E+00	8
6.00E+00	14	-8.00E+00	8
1.20E+01	14	-2.00E+00	2
3.00E+00	14	-1.10E+01	11
6.00E+00	14	-8.00E+00	8
1.20E+01	14	-2.00E+00	2

表中第一列数据代表测试集的预测值，第二列代表实际值，第三第四列代表预测值与实际值之间的误差。对训练集的预测结果如下：

表 6-8：神经网络模型对训练集的预测结果

预测值	实际值	误差	误差绝对值
1.00E+00	14	-1.30E+01	13
3.00E+00	14	-1.10E+01	11
3.00E+00	14	-1.10E+01	11
3.00E+00	14	-1.10E+01	11
3.00E+00	14	-1.10E+01	11
6.00E+00	14	-8.00E+00	8
1.10E+01	14	-3.00E+00	3
1.10E+01	14	-3.00E+00	3
3.00E+00	14	-1.10E+01	11
1.00E+00	14	-1.30E+01	13
1.00E+00	14	-1.30E+01	13
1.10E+01	14	-3.00E+00	3
6.00E+00	14	-8.00E+00	8
3.00E+00	14	-1.10E+01	11
3.00E+00	14	-1.10E+01	11
1.10E+01	14	-3.00E+00	3
1.20E+01	14	-2.00E+00	2
3.00E+00	14	-1.10E+01	11

1.00E+00	14	-1.30E+01	13
1.00E+00	14	-1.30E+01	13

在上表中第一列代表预测值，第二列代表实际值，第三列第四列代表误差。所以经过计算得到平均误差为：0.4769。所以就这一模型的训练集的预测结果与测试集的预测结果相比较而言，说明这一算法模型的适应性比较强，可移植性比较高。不会产生过拟合的状态。就综合几个算法而言，这一算法模型的精度较好。可能是因为隐藏层数较少的原因，但是通过添加更多的隐藏层，算法的时间复杂度和内存占用也会大幅度上升。

综合这几种算法模型来说决策树，kNN，神经网络模型表现在统一水平线上，NB 模型的表现较差，但是 kNN 模型和 NB 模型的时间复杂度较小，运行起来较为简单，但是决策树模型和神经网络模型生成的模型结果较为复杂。

6.5 随机森林模型的构建结果

在森林模型建造的过程中使用了和决策树模型相同的建树参数，森林模型仅仅需要输入特征值和预测值以及建树数量的参数。为了能够看出树的数量对算法预测效果的影响，使用 100 棵决策树建立模型和 1000 棵树建立模型并进行对比。下面是 100 棵树对测试集进行预测的结果：

表 6-9：随机森林模型对训练集的预测结果

预测值	实际值	误差	误差绝对值
5.00E+00	14	-9.00E+00	9
3.00E+00	14	-1.10E+01	11
3.00E+00	14	-1.10E+01	11
4.00E+00	14	-1.00E+01	10
3.00E+00	14	-1.10E+01	11
6.00E+00	14	-8.00E+00	8
5.00E+00	14	-9.00E+00	9
1.00E+01	14	-4.00E+00	4
1.30E+01	14	-1.00E+00	1
4.00E+00	14	-1.00E+01	10
4.00E+00	14	-1.00E+01	10
7.00E+00	14	-7.00E+00	7
1.00E+01	14	-4.00E+00	4

1.10E+01	14	-3.00E+00	3
9.00E+00	14	-5.00E+00	5
5.00E+00	14	-9.00E+00	9
4.00E+00	14	-1.00E+01	10
4.00E+00	14	-1.00E+01	10
4.00E+00	14	-1.00E+01	10
7.00E+00	14	-7.00E+00	7

在上表中第一列代表预测值，第二列代表实际值，第三列第四列代表误差。所以经过计算得到平均误差为：0.4119。再建立第二个拥有 1000 棵树的随机森林结果如下：

表 6-10：随机森林模型对训练集的预测结果

预测值	实际值	误差	误差绝对值
5.00E+00	14	-9.00E+00	9
3.00E+00	14	-1.10E+01	11
3.00E+00	14	-1.10E+01	11
6.00E+00	14	-8.00E+00	8
3.00E+00	14	-1.10E+01	11
1.00E+01	14	-4.00E+00	4
6.00E+00	14	-8.00E+00	8
1.00E+01	14	-4.00E+00	4
1.30E+01	14	-1.00E+00	1
4.00E+00	14	-1.00E+01	10
5.00E+00	14	-9.00E+00	9
7.00E+00	14	-7.00E+00	7
1.00E+01	14	-4.00E+00	4
1.10E+01	14	-3.00E+00	3
9.00E+00	14	-5.00E+00	5
5.00E+00	14	-9.00E+00	9
4.00E+00	14	-1.00E+01	10
6.00E+00	14	-8.00E+00	8
5.00E+00	14	-9.00E+00	9
7.00E+00	14	-7.00E+00	7

在上表中第一列代表预测值，第二列代表实际值，第三列第四列代表误差。所以经过计算得到平均误差为：0.4131。

从总体上来说，随机森林模型的确可以通过建立多个决策树，相互之间通过取众数或平均树为决策树带来很好的结果。但是同时很需要注意的是，拥有 1000 棵树的随机树虽然比 100 棵树拥有更多的树，但是其实并没有为这一模型带来更好的预测效果，这也说明，随机森林模型的表现其实也是并不稳定的。并不是说通过建立更多的树就可以得到更多好的结果。

6.6 集成学习模型的构建结果

在集成学习模型建立的实验中，可以通过不同的分类器来建立集成学习模型，因为已经知道了森林模型的预测能力以及预测效果，所以暂时先不将决策树放入集成学习模型中。多个决策树组合预测能力在实验中已经大致了解了。在这种情形下，再建立集成学习模型，如果建立的模型能够得到更好的效果，如果这时再加入决策树模型，那么决策树模型就很有可能会降低已有的模型的预测效果。同样对获得的集成学习模型进行测试，如果测试得到的效果要弱于随机森林的结果，那么这时候如果在模型中加入决策树，虽然决策树能够然集成学习模型的效果变得更好，但是这样集成学习模型的预测效果也将不会比随机森林的预测效果要好。

所以，首先先将决策树不纳入集成学习模型的建立的过程，建立集成学习模型。集成学习模型首先先通过抽取随机数来进行抽样。在一共 5399 个数据点中选取 540 个作为每一个单个的分类器的训练集，在成功抽样后将抽取的样本点拿来训练模型，一共生成 100 个模型，再用 100 个模型对测试集进行预测，将得到的所有预测值求平均值，最终的值作为结果，并与随机森林的预测结果做出比较。

表 6-11：集成学习模型对训练集的预测结果

预测值	实际值	误差	误差绝对值
6.42E+00	14	-7.58E+00	7.576159
5.65E+00	14	-8.35E+00	8.350993
5.07E+00	14	-8.93E+00	8.930464
3.81E+00	14	-1.02E+01	10.19205
7.63E+00	14	-6.37E+00	6.374172
6.46E+00	14	-7.54E+00	7.543046
4.19E+00	14	-9.81E+00	9.81457
4.41E+00	14	-9.59E+00	9.589404

5.54E+00	14	-8.46E+00	8.460265
6.18E+00	14	-7.82E+00	7.817881
6.51E+00	14	-7.49E+00	7.493377
5.29E+00	14	-8.71E+00	8.705298
4.95E+00	14	-9.05E+00	9.05298
4.78E+00	14	-9.22E+00	9.221854
5.49E+00	14	-8.51E+00	8.509934
4.45E+00	14	-9.55E+00	9.55298
7.00E+00	14	-7.00E+00	6.996689
6.89E+00	14	-7.11E+00	7.10596
4.70E+00	14	-9.30E+00	9.304636
5.41E+00	14	-8.59E+00	8.589404

在上表中第一列代表预测值，第二列代表实际值，第三列第四列代表误差。所以经过计算得到平均误差为：0.3594。

所以说明集成学习模型的预测效果是明显好于随机森林的，为了能够对模型的预测效果进行进一步的探索性提升，因为 NB 模型在单个分类器对结果进行预测时结果是不太好的，所以在实验中将 NB 模型去除，去除之后再一次对结果进行预测预测效果如下：

表 6-12：去除 NB 模型的实验结果

2.00E+00	2.00E+00	2.00E+00
2.00E+00	1.10E+01	3.00E+00
8.00E+00	8.00E+00	1.10E+01
7.00E+00	1.00E+00	7.00E+00
2.00E+00	8.00E+00	5.00E+00
1.30E+01	3.00E+00	4.00E+00
1.00E+01	2.00E+00	1.00E+01
4.00E+00	6.00E+00	3.00E+00
3.00E+00	3.00E+00	8.00E+00
1.30E+01	5.00E+00	4.00E+00
2.00E+00	2.00E+00	4.00E+00
1.00E+01	1.10E+01	3.00E+00
4.00E+00	3.00E+00	3.00E+00
2.00E+00	1.00E+00	1.10E+01
3.00E+00	5.00E+00	4.00E+00
2.00E+00	9.00E+00	4.00E+00

7.00E+00	6.00E+00	7.00E+00
2.00E+00	3.00E+00	2.00E+00
1.00E+01	2.00E+00	8.00E+00
2.00E+00	1.30E+01	1.30E+01

表 6-13：集成学习模型对训练集的预测结果

预测值	实际值	误差	误差绝对值
6.13E+00	14	-7.87E+00	7.865672
6.20E+00	14	-7.80E+00	7.79602
6.21E+00	14	-7.79E+00	7.791045
5.31E+00	14	-8.69E+00	8.691542
6.62E+00	14	-7.38E+00	7.383085
5.62E+00	14	-8.38E+00	8.383085
5.70E+00	14	-8.30E+00	8.303483
6.26E+00	14	-7.74E+00	7.736318
6.13E+00	14	-7.87E+00	7.870647
5.82E+00	14	-8.18E+00	8.179104
5.98E+00	14	-8.02E+00	8.0199
5.97E+00	14	-8.03E+00	8.029851
6.35E+00	14	-7.65E+00	7.651741
5.70E+00	14	-8.30E+00	8.298507
5.97E+00	14	-8.03E+00	8.034826
6.10E+00	14	-7.90E+00	7.895522
6.88E+00	14	-7.12E+00	7.124378
6.14E+00	14	-7.86E+00	7.855721
6.00E+00	14	-8.00E+00	8.004975
7.00E+00	14	-7.00E+00	7

在上表中第一列代表预测值，第二列代表实际值，第三列第四列代表误差。所以经过计算得到平均误差为：0.3514。

所以得到了比上一个集成学习模型更好的结果。

通过单独建立 NB 模型的集成学习模型得到的结果为

表 6-14：NB 集成学习模型对训练集的预测结果

预测值	实际值	误差	误差绝对值
7.00E+00	14	-7.00E+00	7

4.54E+00	14	-9.46E+00	9.455446
2.80E+00	14	-1.12E+01	11.19802
8.22E-01	14	-1.32E+01	13.17822
9.63E+00	14	-4.37E+00	4.366337
8.13E+00	14	-5.87E+00	5.871287
1.18E+00	14	-1.28E+01	12.82178
7.23E-01	14	-1.33E+01	13.27723
4.37E+00	14	-9.63E+00	9.633663
6.90E+00	14	-7.10E+00	7.09901
7.55E+00	14	-6.45E+00	6.445545
3.95E+00	14	-1.00E+01	10.0495
2.16E+00	14	-1.18E+01	11.84158
2.94E+00	14	-1.11E+01	11.05941
4.54E+00	14	-9.46E+00	9.455446
1.15E+00	14	-1.29E+01	12.85149
7.26E+00	14	-6.74E+00	6.742574
8.39E+00	14	-5.61E+00	5.613861
2.11E+00	14	-1.19E+01	11.89109
2.25E+00	14	-1.18E+01	11.75248

在上表中第一列代表预测值，第二列代表实际值，第三列第四列代表误差。所以经过计算得到平均误差为：0.4344。

通过 NB 模型建立的集成学习模型可以说明，单个的预测效果不那么满意的分类器，也可以通过集成学习模型来让分类器的结果有大幅度的提升。

在建立集成学习模型的过程中，实验建立的集成学习模型通过训练和预测得到了比随机森林要更加好的效果。这一种集成学习模型有更好的表现的原因可能是在这一组训练集上，这几个分类器相较于决策树分类器来说更容易受一些随机因素的影响，所以当在这三个模型上使用集成学习模型时，模型有了很大程度的提升。

7 后期研究计划

目前在此实验中，实验通过建立集成学习模型得到了相较于随机森林模型更好的结果。但是这一现象背后的理论原因尚未探明，以及现在的实验只是在比较小的数据集上对这一方法进行预测并取得了结果。所以，在未来的工作中，首先需要扩大实验的数据集，通过扩大数据集来让实验在这一问题上能更具有普适性。然后需要对实验进行过程中的各个过程进行调整和细化。现在的实验中，使用 1000 作为时间窗口的大小，但是不同大小的时间窗口相较于不同的算法模型可能也会带来不同的效果，所以需要调整时间窗口的大小继续进行实验。并且针对与不同的模型，比如决策树模型，kNN 模型，朴素贝叶斯和神经网络模型，这些模型在这个实验的使用过程中也不一定达到了分别实验的最好效果，所以需要对不同的实验模型进行分别的探究，了解怎么样才能达到某一个模型的最佳的预测效果。在次基础上然后在再次进行集成学习模型的构建，希望能够达到更好的效果。

8 结　论

这一次实验通过数据分析模型对实验室地震数据进行了分析，在实验数据的其中一部分数据集上，我们可以发现，当使用单个的分类方法时，决策树的预测效果是最好的，依次是最近邻模型，神经网络模型和朴素贝叶斯模型，可能是因为在描述特征值与目标值的过程中，决策树更能很好的描述相对复杂的关系，所以能够得到更好的结果。但是就时间复杂度来说，当样本的数据点增加，或者是添加更多的特征用于数据集的预测时，决策树模型和神经网络模型所需要的建立模型的时间也就越长。但是朴素贝叶斯模型就可以以相对更少的时间完成模型的建立和预测。在这一问题中，在决策树模型上使用集成学习模型，也就是随机森林模型并没有为整个实验的预测带来更好的结果，但是其余三个模型在这个问题上，因为分析的相对简单，建立的模型也很简单，所以在实际预测过程中更容易收到一些随机因素的影响，所以在这三个模型上使用集成学习模型时，模型的预测效果会有大幅度的提升。

参考文献

[1] Alexandridis A, Chondrodima E, Efthimiou E, et al. Large earthquake occurrence estimation based on radial basis function Neural Networks[J]. IEEE Transactions on Geoscience and Remote Sensing, 2014, 52(9):5443-5453.http://www.seg.org /reviews/mcc orm30.html.

[2] Rouet-Leduc B, Hulbert C, Lubbers N, et al. Machine Learning Predicts Laboratory Earthquakes[J]. Geophysical Research Letters, 2017, 44(18).

[3] Schwartz D P, Coppersmith K J . Fault behavior and characteristic earthquakes: Examples from the Wasatch and San Andreas Fault Zones[J]. Journal of Geophysical Research, 1984, 89(B7):5681.

[4] 尹祥础. 地震预测新途径的探索[J]. 中国地震, 1987(1):3-10.

[5] Johnson P A, Ferdowsi B, Kaproth B M, et al. Acoustic emission and microslip precursors to stick-slip failure in sheared granular material[J]. Geophysical Research Letters, 2013, 40(21):5627-5631.

[6] 唐荣余, 陈连旺. 地震科技发展战略的思考——建议组建地震数值模拟实验室[J]. 国际地震动态, 2005(7):15-16.

[7] 科美. 实验室模拟地震[J]. 安全与健康, 2003(7):50-50.

[8] King C Y, 谢觉民. 实验室断层研究显示的地震机制及其可预报性[J]. 地震地质译丛, 1993(1):3-13.

[9] Rouet-Leduc B, Hulbert C, Bolton D C, et al. Estimating Fault Friction from Seismic Signals in the Laboratory[J]. 2017, 45(3).

[10] Leeman J R, Saffer D M, Scuderi M M, et al. Laboratory observations of slow earthquakes and the spectrum of tectonic fault slip modes[J]. Nature Communications, 2016, 7:11104.

[11] Quinlan J R . Induction of decision trees[J]. Machine Learning, 1986, 1.

[12] Shannon C E . A Mathematical Theory of Communication[J]. The Bell System Technical Journal, 1948, 27.

[13] Durkin J, 蔡竞峰, 蔡自兴. 决策树技术及其当前研究方向[J]. 控制工程, 2005(1) .

[14] Swami A, Jain R . Scikit-learn: Machine Learning in Python[J]. Journal of Machine Learning Research, 2012, 12(10):2825-2830.

[15] Cover T M, Hart P E . Nearest Neighbor Pattern Classification[J]. IEEE Transactions on Information Theory, 1967, 13(1):21-27.

[16] Pereira F, Mitchell T, Botvinick M . Machine learning classifiers and fMRI:

A tutorial overview[J]. NeuroImage, 2009, 45:199-209.

[17] Breiman L . Random Forests[J]. Machine Learning, 2001, 45(1):5-32.

[18] 方匡南, 吴见彬, 朱建平, et al. 随机森林方法研究综述[J]. 统计与信息论坛, 2011(3):32-38.

[19] 姚登举, 杨静, 詹晓娟. 基于随机森林的特征选择算法[J]. 吉林大学学报(工学版), 2014(01).

[20] Kuncheva L I, Whitaker C J. Measures of Diversity in Classifier Ensembles and Their Relationship with the Ensemble Accuracy[J]. Machine Learning, 2003, 51(2):181-207.

[21] Dietterich T G. Ensemble learning[J]. The handbook of brain theory and neural networks, 2002, 2: 110-125.

[22] Zaharia M, Chowdhury M, Franklin M J, et al. Spark: Cluster computing with working sets[J]. HotCloud, 2010, 10(10-11): 95.

[23] Chai T, Draxler R R. Root mean square error (RMSE) or mean absolute error–Arguments against avoiding RMSE in the literature[J]. Geoscientific model development, 2014, 7(3): 1247-1250.

[24] Marzuki Z, Ahmad F. Data mining discretization methods and performances[J]. lung, 2012, 3(32): 57.

[25] Vahid Mirjalili, Sebastian Raschka . Python Machine Learning: Perform Python Machine Learning and Deep Learning with Python, scikit-learn, and TensorFlow[M]. Packt Publishing, 2017.

[26] James Yan, Dr. Yuxing Yan . Hands-On Data Science with Anaconda[M]. Packt Publishing, 2018.

[27] Aurélien Géron . Hands-On Machine Learning with Scikit-Learn and TensorFlow[M]. O'Reilly Media, Inc, 2017.

附录A 部分 Python 以及 Spark 关键代码

一、 数据预处理 Spark 代码：

```
from pyspark.sql import SparkSession

spark = SparkSession \
    .builder \
    .appName ( "Python Spark SQL basic" )    \
    .config("spark.some.config.option", "some-value") \
    .getOrCreate()


spark
```

```
from pyspark.sql import Row
from pyspark.sql.types import *
from pyspark.sql import SQLContext
import time

kaishi = time.clock()

sc = spark.sparkContext
sqlContext = SQLContext(sc)

#创建DataFrame
HouseInfo =
sqlContext.read.csv(r"E:\EARTHQUAKE\LANL-Earthquake-Prediction\train.csv",heade
r = True, inferSchema = True)
#检测是否创建成功
type ( HouseInfo )

jieshu = time.clock( )
print ( jieshu - kaishi )


import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
dataset =
pd.read_csv('E:\EARTHQUAKE\LANL-Earthquake-Prediction\\train.csv',skiprows=500
000,nrows = 600000,index_col = 1)
x = np.linspace(0,22500,600000)
plt.plot(x,dataset)
```

二、 时间窗口：

```
import numpy as np
import pandas as pd
```

```
import time

import csv

kaishi = time.clock()

#窗口大小
#1000
win = int(1000)
#波形个数
#5
wave = 1
#读取行数
wave_handled = wave*5400000
#有多少个时间窗口
win_num = int(wave_handled/win)
numbers = np.linspace(1,win_num,win_num)
TreeData = np.zeros((win_num,3))
target = np.zeros((win_num,1))
for num in numbers:
    i = int(num)
    Earthquake_t =
pd.read_csv('E:\EARTHQUAKE\LANL-Earthquake-Prediction\\train.csv',skiprows =
(i-1)*win,nrows = win,index_col = 0)
    target_lable = int(np.median(Earthquake_t)*10)
    target[i-1] = target_lable
#print(target)
for num in numbers:
    i = int(num)
    Earthquake_a =
pd.read_csv('E:\EARTHQUAKE\LANL-Earthquake-Prediction\\train.csv',skiprows =
(i-1)*win,nrows = win,index_col = 1)
    array = Earthquake_a
    TreeData[i-1][0] = np.mean(array)
##      print(TreeData[i-1][0])
    TreeData[i-1][1] = np.var(array)
##      print(TreeData[i-1][1])
    TreeData[i-1][2] = np.max(array)-np.min(array)
##      print(TreeData[i-1][2])
#print(TreeData)

processedData = np.c_[TreeData,target]

with open('train.csv','w') as csvFile:
    writer = csv.writer(csvFile)
    writer.writerows(processedData)

jieshu = time.clock()

print(jieshu-kaishi)

import numpy as np
import pandas as pd

import time
```

```
import csv

kaishi = time.clock()

#窗口大小
#1000
win = int(1000)
#波形个数
#5
wave = 1
#读取行数
wave_handled = wave*5400000
#有多少个时间窗口
win_num = int(wave_handled/win)
numbers = np.linspace(1,win_num,win_num)
TreeData = np.zeros((win_num,3))
target = np.zeros((win_num,1))
for num in numbers:
    i = int(num)
    Earthquake_t =
pd.read_csv('E:\EARTHQUAKE\LANL-Earthquake-Prediction\\train.csv',skiprows =
44500000+(i-1)*win,nrows = win,index_col = 0)
    target_lable = int(np.median(Earthquake_t)*10)
    target[i-1] = target_lable
#print(target)
for num in numbers:
    i = int(num)
    Earthquake_a =
pd.read_csv('E:\EARTHQUAKE\LANL-Earthquake-Prediction\\train.csv',skiprows =
44500000+(i-1)*win,nrows = win,index_col = 1)
    array = Earthquake_a
    TreeData[i-1][0] = np.mean(array)
##      print(TreeData[i-1][0])
    TreeData[i-1][1] = np.var(array)
##      print(TreeData[i-1][1])
    TreeData[i-1][2] = np.max(array)-np.min(array)
##      print(TreeData[i-1][2])
#print(TreeData)

processedData = np.c_[TreeData,target]

with open('test.csv','w') as csvFile:
    writer = csv.writer(csvFile)
    writer.writerows(processedData)

jieshu = time.clock()
print(jieshu-kaishi)
```

三、 决策树建立模型：

```
import numpy as np
```

```
import pandas as pd
from sklearn.tree import DecisionTreeClassifier
from sklearn.tree import export_graphviz
from sklearn import tree

TreeData = pd.read_csv("E:\BSDATA\\train_X.csv")
target = pd.read_csv("E:\BSDATA\\train_target.csv")
test_TreeData = pd.read_csv("E:\BSDATA\\1.csv")
test_target = pd.read_csv("E:\BSDATA\\2.csv")
test_target = test_target.values

tree_clf = DecisionTreeClassifier()
tree_clf.fit(TreeData,target)
result = tree_clf.predict(test_TreeData)


np.savetxt('E:\BSDATA\DE_RES\\R.csv',result,delimiter = ',')
```

四、 kNN 建立模型

```
import numpy as np
import pandas as pd
from sklearn.neighbors import KNeighborsClassifier

TreeData = pd.read_csv("E:\BSDATA\\train_X.csv")
target = pd.read_csv("E:\BSDATA\\train_target.csv")
target = np.array(target)
test_TreeData = pd.read_csv("E:\BSDATA\\1.csv")
test_target = pd.read_csv("E:\BSDATA\\2.csv")

knn = KNeighborsClassifier(n_neighbors = 15)
knn.fit(TreeData,target.ravel())

result = knn.predict(TreeData)
np.savetxt('E:\BSDATA\kNN_RES\\R.csv',result,delimiter = ',')
```

五、 NB 建立模型

```
import numpy as np
import pandas as pd
from sklearn.naive_bayes import GaussianNB

TreeData = pd.read_csv("E:\BSDATA\\train_X.csv")
target = pd.read_csv ( "E:\BSDATA\\train_target.csv")
target = np.array ( target)
test_TreeData = pd.read_csv ( "E:\BSDATA\\1.csv")
test_target = pd.read_csv ( "E:\BSDATA\\2.csv")

gnb = GaussianNB ( )
gnb.fit ( TreeData,target.ravel ( ))

result = gnb.predict ( test_TreeData)
np.savetxt ( 'E:\\BSDATA\\NB_RES\\result.csv',result,delimiter = ',')
```

六、 神经网络建立模型

```
import numpy as np
import pandas as pd
from sklearn.neural_network import MLPClassifier

TreeData = pd.read_csv ( "E:\BSDATA\\train_X.csv")
target = pd.read_csv ( "E:\BSDATA\\train_target.csv")
target = np.array ( target)
test_TreeData = pd.read_csv ( "E:\BSDATA\\1.csv")
test_target = pd.read_csv ( "E:\BSDATA\\2.csv")

clf   =   MLPClassifier ( solver = 'lbfgs', alpha = 1e-5,
                          hidden_layer_sizes =   ( 5, 2), random_state = 15)
clf.fit ( TreeData,target.ravel ( ))

clf.predict ( test_TreeData)
result = clf.predict ( test_TreeData)
np.savetxt ( 'E:\BSDATA\\N_RES\\result.csv',result,delimiter = ',')
```

七、 随机森林建立模型

```
import numpy as np
import pandas as pd
from sklearn.ensemble import RandomForestClassifier

TreeData = pd.read_csv ( "E:\BSDATA\\train_X.csv")
target = pd.read_csv ( "E:\BSDATA\\train_target.csv")
target = np.array ( target)
test_TreeData = pd.read_csv ( "E:\BSDATA\\1.csv")
test_target = pd.read_csv ( "E:\BSDATA\\2.csv")

clf   =   RandomForestClassifier ( n_estimators = 1000)
clf.fit ( TreeData,target.ravel ( ))

clf.predict ( test_TreeData)
result = clf.predict ( test_TreeData)
np.savetxt ( 'E:\BSDATA\\RF_RES\\result1000.csv',result,delimiter = ',')
```

八、 集成学习模型的建立

```
import numpy as np
import pandas as pd
import random

from sklearn.neighbors import KNeighborsClassifier

numbers = np.linspace ( 1,100,100)
```

```
numbers1 = np.linspace ( 1,540,540)


RowData = pd.read_csv ( "E:\BSDATA\\train_X.csv")
RowData = np.array ( RowData)
Rowtarget = pd.read_csv ( "E:\BSDATA\\train_target.csv")
Rowtarget = np.array ( Rowtarget)
test_TreeData = pd.read_csv("E:\BSDATA\\1.csv")

TreeData = np.zeros((540,3))
target = np.zeros((540,1))
R = np.zeros((5399,1))

for num in numbers:
    test_num = random.sample(range(1,5399),540)
    test_num.sort()
    for num1 in numbers1:
        i = int(num1)
        TreeData[i-1,0] = RowData[test_num[i-1],0]
        TreeData[i-1,1] = RowData[test_num[i-1],1]
        TreeData[i-1,2] = RowData[test_num[i-1],2]
        target[i-1,0] = Rowtarget[test_num[i-1],0]

    knn = KNeighborsClassifier ( n_neighbors = 15)
    knn.fit ( TreeData,target.ravel ( ))
    result = knn.predict ( test_TreeData)
    result.shape =    ( 5399,1)
    R = np.c_[R,result]



np.savetxt('E:\BSDATA\JC_RES\\KNNR.csv',R,delimiter = ',')


import numpy as np
import pandas as pd
import random

from sklearn.naive_bayes import GaussianNB

numbers = np.linspace(1,100,100)
numbers1 = np.linspace(1,540,540)


RowData = pd.read_csv("E:\BSDATA\\train_X.csv")
RowData = np.array(RowData)
Rowtarget = pd.read_csv("E:\BSDATA\\train_target.csv")
Rowtarget = np.array(Rowtarget)
test_TreeData = pd.read_csv("E:\BSDATA\\1.csv")

TreeData = np.zeros((540,3))
target = np.zeros((540,1))
R = np.zeros((5399,1))
```

```
for num in numbers:
    test_num = random.sample(range(1,5399),540)
    test_num.sort()
    for num1 in numbers1:
        i = int(num1)
        TreeData[i-1,0] = RowData[test_num[i-1],0]
        TreeData[i-1,1] = RowData[test_num[i-1],1]
        TreeData[i-1,2] = RowData[test_num[i-1],2]
        target[i-1,0] = Rowtarget[test_num[i-1],0]

    gnb = GaussianNB()
    gnb.fit(TreeData,target.ravel())
    result = gnb.predict(test_TreeData)
    result.shape = (5399,1)
    R = np.c_[R,result]



np.savetxt('E:\BSDATA\JC_RES\\NBR.csv',R,delimiter = ',')


import numpy as np
import pandas as pd
import random

from sklearn.neural_network import MLPClassifier

numbers = np.linspace ( 1,100,100)
numbers1 = np.linspace ( 1,540,540)


RowData = pd.read_csv ( "E:\BSDATA\\train_X.csv")
RowData = np.array ( RowData)
Rowtarget = pd.read_csv ( "E:\BSDATA\\train_target.csv")
Rowtarget = np.array ( Rowtarget)
test_TreeData = pd.read_csv ( "E:\BSDATA\\1.csv")

TreeData = np.zeros (    ( 540,3))
target = np.zeros (    ( 540,1))
R = np.zeros (    ( 5399,1))

for num in numbers:
    test_num = random.sample ( range ( 1,5399),540)
    test_num.sort ( )
    for num1 in numbers1:
        i = int ( num1)
        TreeData[i-1,0] = RowData[test_num[i-1],0]
        TreeData[i-1,1] = RowData[test_num[i-1],1]
        TreeData[i-1,2] = RowData[test_num[i-1],2]
        target[i-1,0] = Rowtarget[test_num[i-1],0]

    clf   =   MLPClassifier ( solver = 'lbfgs', alpha = 1e-5,
                              hidden_layer_sizes =    ( 5, 2), random_state = 15)
    clf.fit ( TreeData,target.ravel ( ))
```

```
    result = clf.predict ( test_TreeData)
    result.shape =    ( 5399,1)
    R = np.c_[R,result]


np.savetxt ( 'E:\BSDATA\JC_RES\\NR.csv',R,delimiter = ',')
```

在学取得成果

九、 在学期间所获的奖励

十、 在学期间发表的论文

十一、 在学期间取得的科技成果

2017 年 11 月，以 UST_Beijing 队数学建模队员的身份参加 iGEM 比赛并获得铜牌。

2018 年 3 月，以队员身份参加摇篮杯比赛并获得二等奖。

致　谢

这是我第一次在导师的指导下独立自主地完成一篇学科课题的论文，感谢我的导师储继迅老师在此期间对我的指导和帮助. 储老师在我完成毕业设计的过程中给予了我无数的关心和帮助，在毕业设计初期教给我查阅论文的方式和方法，在开题和中期检查时对我们一再强调论文的格式，要保持严谨的态度，包括在最后当我的程序需要运行时，储老师帮助我联系老师找大型计算机。储老师在我做毕业设计的过程中用数学思想不断启发我，让我收获的不仅仅是数学方面的知识和方法，更多的是某种内在的学术气质和精神，能让我将来在我的学术生涯中受益。

在完成论文的过程中，我也要感谢各种软件的开发者，正是因为他们的无私奉献我才能够使用到很多很成熟的软件和算法，来完成我的实验和论文。

最后我要感谢所有在我遇到难题时曾经帮助过我的老师和同学，以及在本文中被我引用或参考的论著的作者。