机器学习大作业技术报告

18231217 吴陶然

一. 医疗花费预测

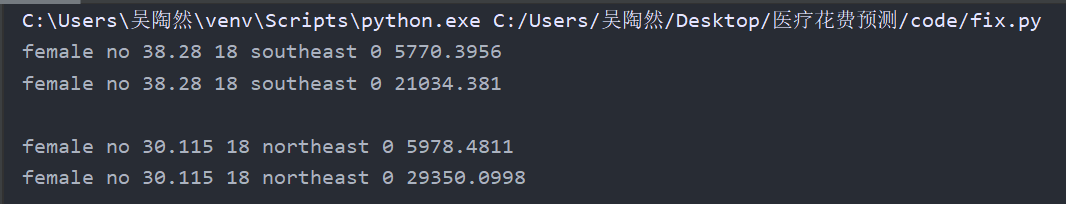
1. 方法介绍

分别通过全手写不调包实现随机森林、全手写不调包实现线性回归、借助scikit-learn包实现GBDT、SVR、LassoRegression、决策树模型来求解该问题，使用randomizedSearchCV、GridSearchCV、手动调参三种方式进行模型调参，并对RandomForest、GBDT、DecisionTree、SVR、LinerRegression模型进行了模型融合，模型融合时尝试使用了直接平均法、加权平均法、stacking堆叠法进行，同时使用K折交叉验证、留一法等多种方法进行模型评判。

2. 方法实现细节和模型结构

2.1 数据处理和特征工程

读入数据集以后，通过describe方法观察是否数据的大概情况，使用isnull().sum()方法观察发现数据中不存在空值。测试集较小，仅1070行，由于需要尽可能地利用数据，无需舍去数据、无需采样。但在进行检查时发现，数据中存在相同值。通过将数据中sex、smoker、bmi、age、region、children属性均相同，且charges不同的数据输出如下：



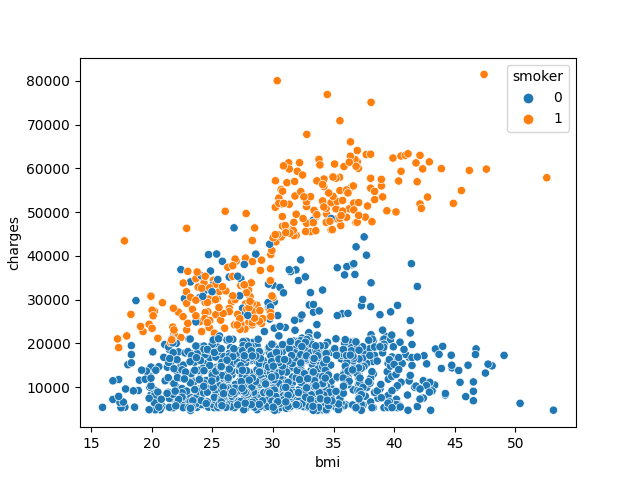
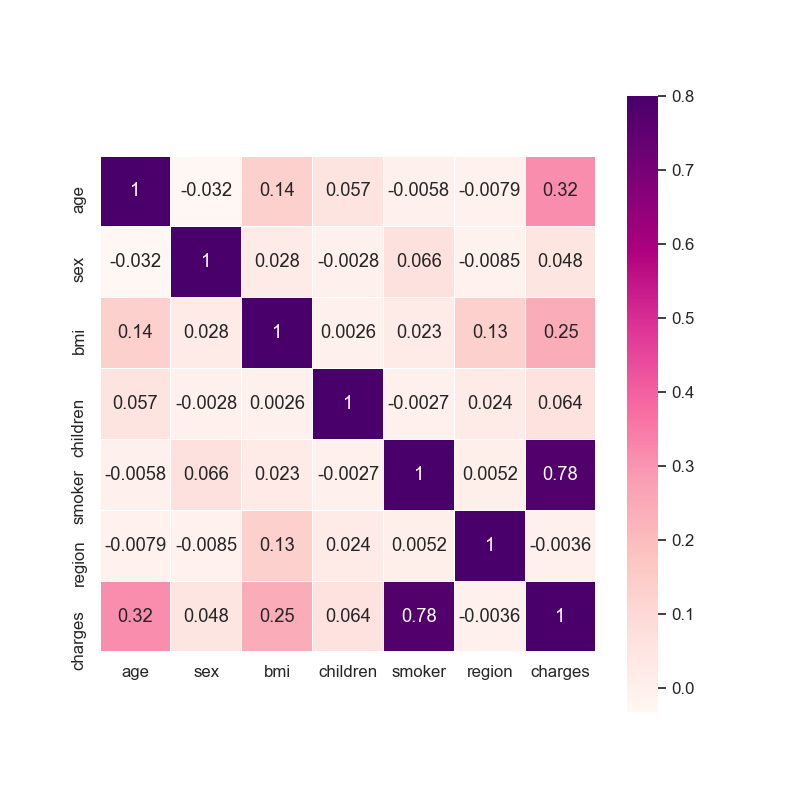




通过训练一个准确率较高的基本模型（此处选取随机森林），使用模型来对这些重复数据的特征进行训练。将charges与模型预测结果差别较大的值当做异常值舍去。

对数据做一些可视化容易得知，bmi对于charges的影响大小，与是否吸烟密切相关。吸烟与否，bmi对于是否charges的影响不同，不吸烟时charges整体较小，吸烟时，当bmi小于30时charges较小，当bmi大于30时charges较大。由于该题特征较少数据量也并不大，故考虑将bmi与smoker融合生成一个新的特征smoker\_bmi。当smoker为0时，smoker\_bmi也为0；当smoker为1，bmi小于30时，smoker\_bmi为1；当smoker为1，bmi大于30时，smoker\_bmi为2。

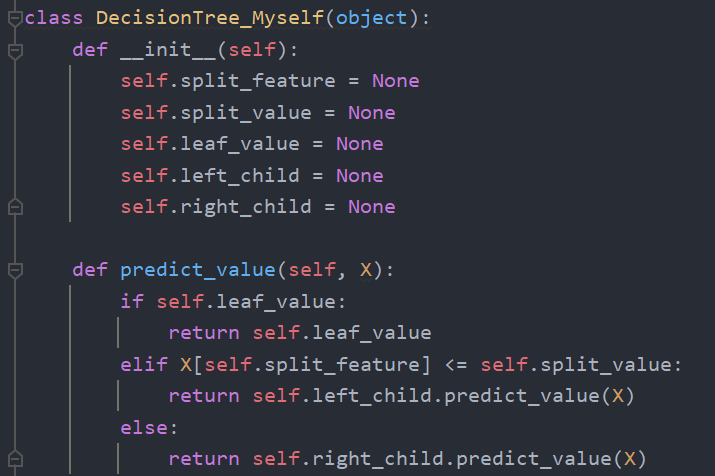
将sex、region、smoker的标签值转为int型，便于训练。



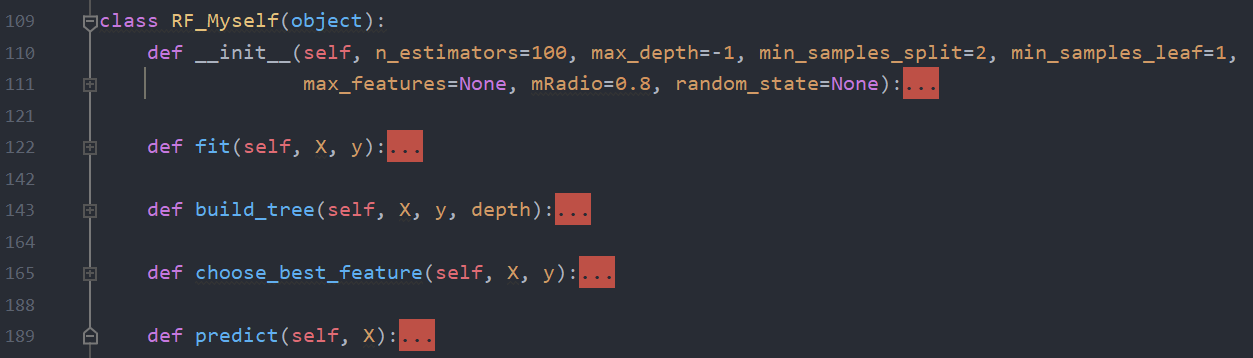


2.2 全手写不调包实现随机森林

首先定义一个决策树的结点类，便于随机森林中建树。决策树结点类共包含五个属性，分别表示该结点的分割特征、分割的阈值、左孩子和右孩子、叶子结点的值（仅当该结点为叶子结点时有效）。同时定义一个方法用于查询某个数据对应的叶子结点值。



接下来定义随机森林类，该类使用了和sklearn里RandomForestRegressor库一样的接口，便于使用。共包含7个参数、tree属性和feature\_importances属性，fit方法用于训练模型，build\_tree方法用于在训练模型时建立决策树，choose\_best\_feature用于在建立决策树时选择最合适的特征进行分裂，predict为模型预测。参数一共包含7个，其中n\_estimators为决策树的数量，默认为100；max\_depth为决策树最大深度限制，默认为-1，即不限制决策树最大深度；min\_samples\_split为决策树分裂时所需要的最小样本数，默认为2；min\_samples\_leaf为叶子结点的最少样本数，默认为1；max\_features为特征的采样方式，有None，sqrt，log2三种，当选择sqrt时，选择的特征数为int(len(X.columns) \*\* 0.5)；当选择log2时，选择的特征数为int(math.log(len(X.columns)))。默认为None，即使用所有特征，不采样。mRadio为行采样比例，默认为0.8。random\_state为随机数种子，默认为None。



在fit方法中，在行采样和列采样后，进行循环建树。通过调用build\_tree方法，循环建立n\_estimators棵决策树，并加决策树加入tree队列中。在build\_tree方法中，只要满足分裂条件（即当前样本数量大于min\_samples\_split、分裂后叶子结点样本数量大于min\_samples\_leaf，决策树深度小于max\_depth），就调用choose\_best\_feature方法选择最合适的特征并选取分裂阈值进行分裂，同时继续递归调用build\_tree。在choose\_best\_feature方法中，通过计算信息增益，选择信息增益最大的特征及阈值并返回，同时会记录选择的特征，并使该特征的重要程度加一，便于记录特征的重要性。

模型建立完成以后，可通过调用predict方法进行预测。在predict方法中，会对tree序列中的每一个决策树进行调用，将每一棵树的结果取平均值，作为随机森林的预测结果。

通过循环遍历参数，选择出合适的参数后，手写实现的随机森林最高能达到0.845的评分。

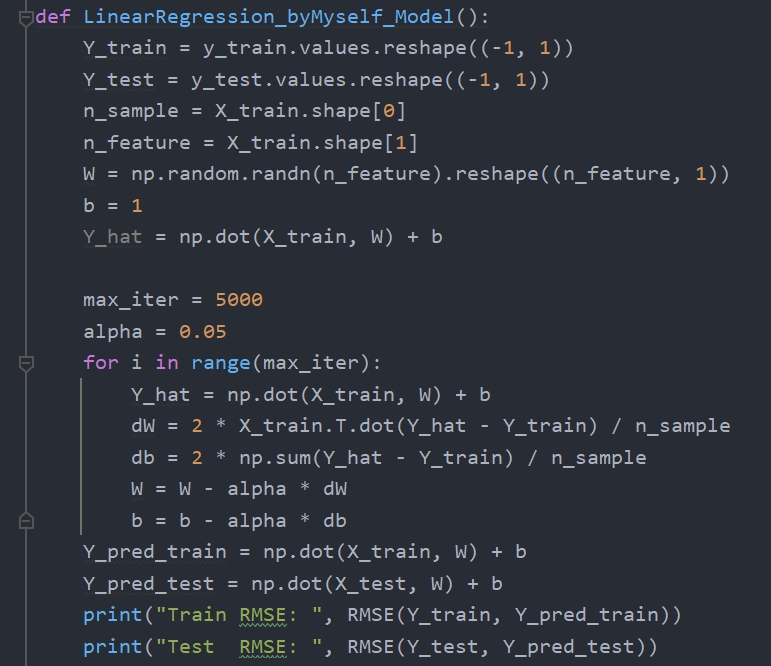


2.3 全手写不调包实现线性回归

模型表达式为：

参数的迭代更新公式为：

对模型进行5000次迭代，同时将学习率设置为0.05。



此外，还需要对数据进行归一化：

对数据进行了归一化以后，整个数据集上的梯度分布能得到改良，有利于进行线性回归建模。在进行调参以后，手写实现线性回归能达到的最佳分数为0.746



2.4 借助scikit-learn包实现GBDT、SVR、LassoRegression、决策树

sklearn包实现GBDT、SVR、LassoRegression、决策树模型，均只需要直接调用模型即可，实现过程较为简单。

对于GBDT而言，调用sklearn包中的GradientBoostingRegressor库进行训练，在经过调参以后GBDT模型准确率能达到0.860。



对于SVR而言，调用sklearn包中的SVR()库进行训练，在经过调参以后SVR模型准确率能达到0.848。



对于LassoRegression而言，调用sklearn包中的Lasso()模型后，在经过调参以后准确率能达到0.742，模型准确率并不高。

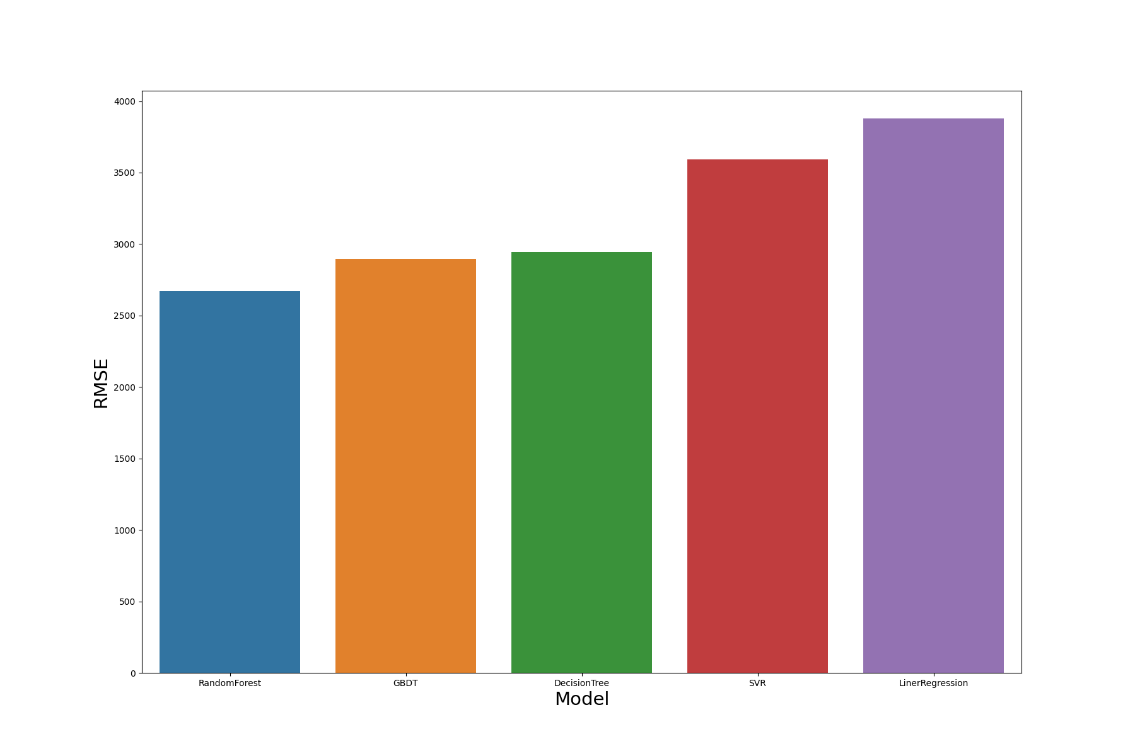


对于决策树模型，调用sklearn包中的DecisionTreeRegresso方法进行训练，在经过调参以后准确率能达到0.848。



2.5 模型融合——直接平均

在各个模型实现以后，几个主要模型验证集上预测结果的均方误差如下所示：



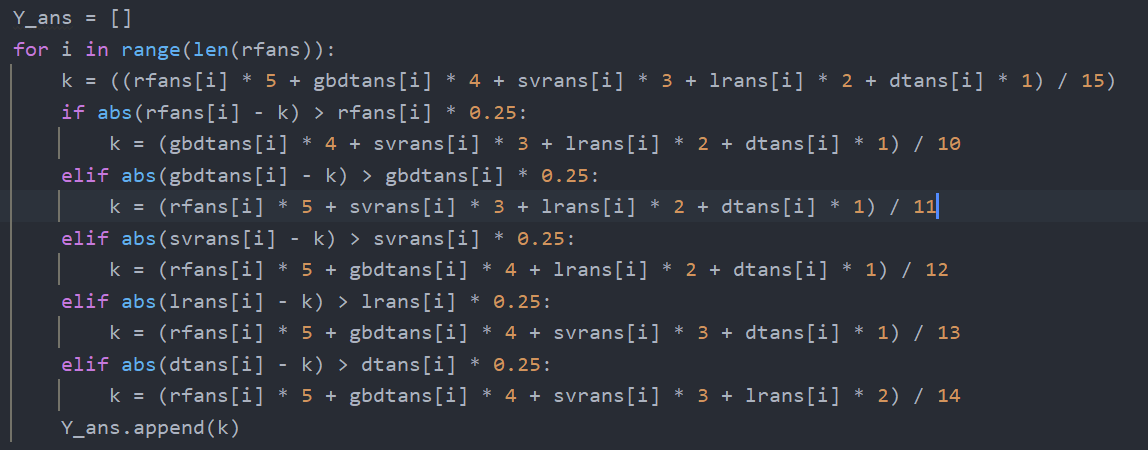
故在进行模型融合时，融合RandomForest、GBDT、DecisionTree、SVR、LinerRegression。取上述模型评分最高的一次预测结果，进行直接平均后输出。出人意料地，最终结果评分仅有0.82。

2.6 模型融合——加权平均及改进尝试

按照加权平均的思想，把各个模型按照性能排名，分别为RandomForest、GBDT、DecisionTree、SVR、LinerRegression模型赋予5、4、3、2、1的权重，对各个模型进行加权平均，输出结果，而该结果仍旧低于五个模型中评分最高的randomforest模型，仅有0.85左右的评分。

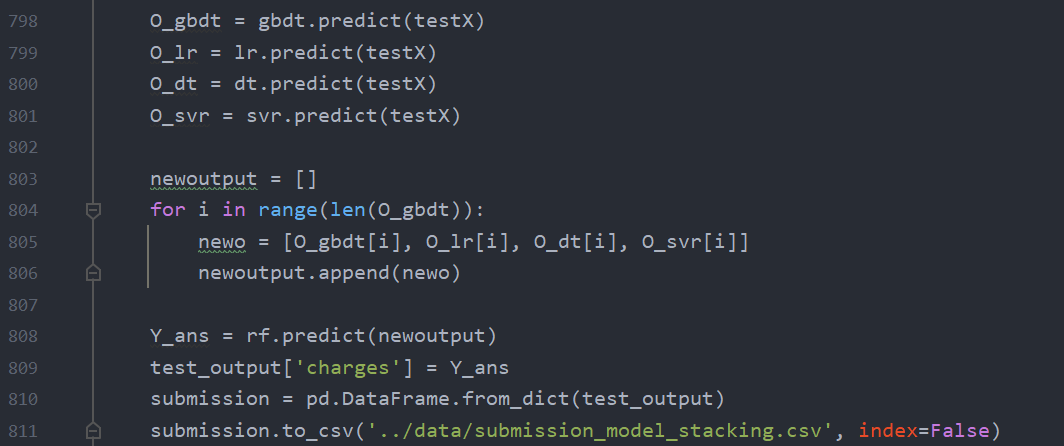
对原因进行探究时发现，相似的模型家家相似，而不同的模型非常不同。如果某个模型预测值与正确值相差过大，则会导致整个预测结果受其影响，虽然进行了加权平均，但也依旧无法抵消这种影响。例如，其余四个模型的预测结果均为8000左右，但另一个模型预测结果为20000，则会导致结果误差较大。

故尝试对模型融合进行改进，根据查阅得到的方法知，对于五个模型融合时，如果某一个模型的预测结果与五个模型预测结果均值的差异在自身25%以上时，就舍去该值不进行模型融合，并重新计算均值结果。



2.7 模型融合——stacking堆叠法

将模型分为两层，第一层初级学习器包括GBDT、线性回归、支持向量机和决策树四种模型，使用这四种模型分别去训练，并将这四种模型在训练集上的输出作为次级学习器的输入，将训练集的charges作为次级学习器的结果，去训练次级学习器，其中次级学习器为randomforest。在进行测试时，分别用上述四个模型去跑测试集的输入，并将其预测结果作为随机森林的输入进行预测。最终随机森林的预测结果即为输出。通过stacking模型融合的效果是最好的。

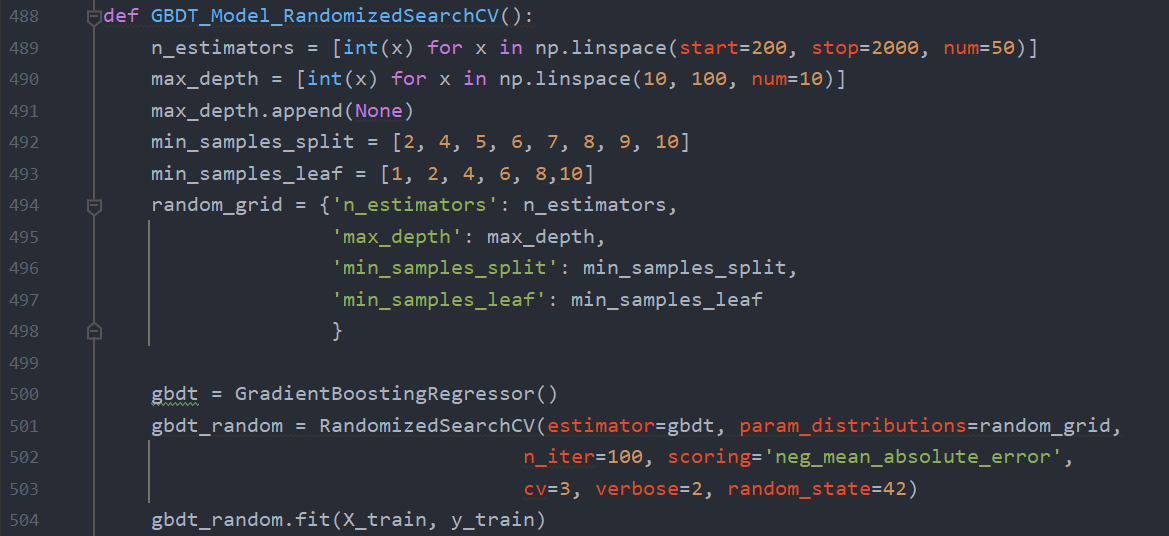


3. 调参过程与模型评判

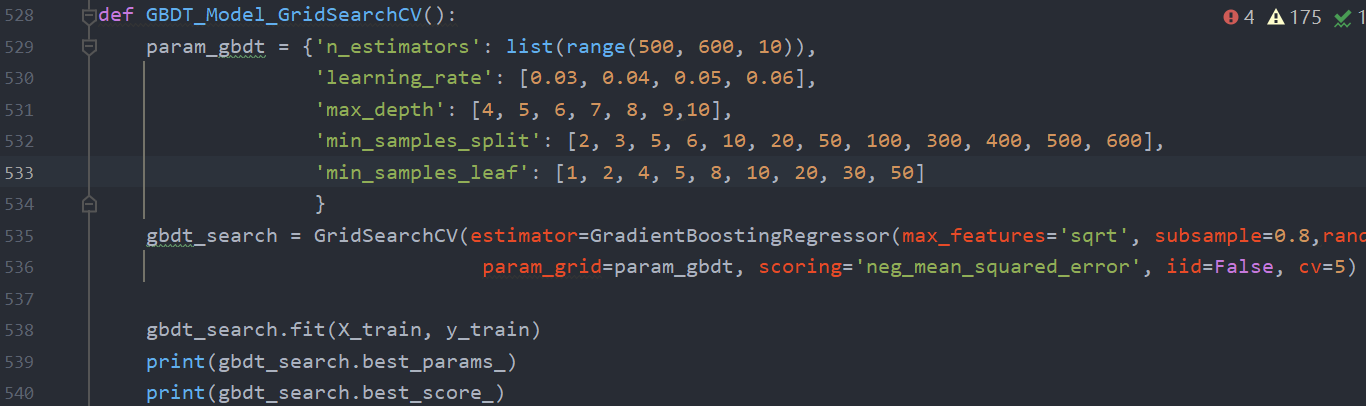
3.1 randomizedSearchCV、GridSearchCV进行自动调参

使用randomizedSearchCV、GridSearchCV进行自动调参。

例如使用randomizedSearchCV对GBDT进行调参时，调节的参数有n\_estimators、max\_depth、min\_samples\_split、min\_samples\_leaf，调节的范围如下所示：



使用GridSearchCV对GBDT调参，调节的参数有n\_estimators，learning\_rate，max\_depth，min\_samples\_split，min\_samples\_leaf，调节的范围如下：

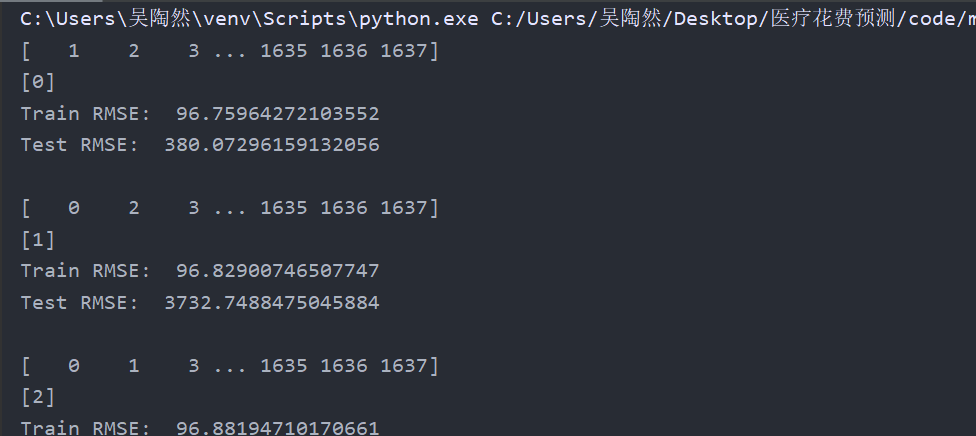


调参结束以后，可以通过输出best\_estimator\_属性来查看最佳参数。

3.2 K折交叉验证和留一法

由于本题采用了很多个模型，故对于模型的评判至关重要。采用k折交叉验证（k取5）和留一法进行模型的评判，通过记录模型预测结果的均方根误差，取所有预测结果均方根误差的平均值。

因为这题数据量比较小，运行时间并不会很长，所以可直接使用留一法进行验证。因为验证次数很多，所以留一法相对比较准确，但是计算量很大。



4. 遇到的问题和解决方法

遇到的问题一：手写实现随机森林时若建立决策树较多，随机森林运行速率较慢

解决方法：通过学习sklearn库的RandomForestRegressor源代码可知，可以将循环建立决策树更改为使用parallel进行并行建立决策树，在一定情况下能大幅提高效率。

遇到的问题二：手写实现线性回归，运行结果的方差过大，结果不准确

解决方法：线性回归的学习率不能设置得过大，当学习率设置得过大时，会出现震荡现象，使得误差越来越大。

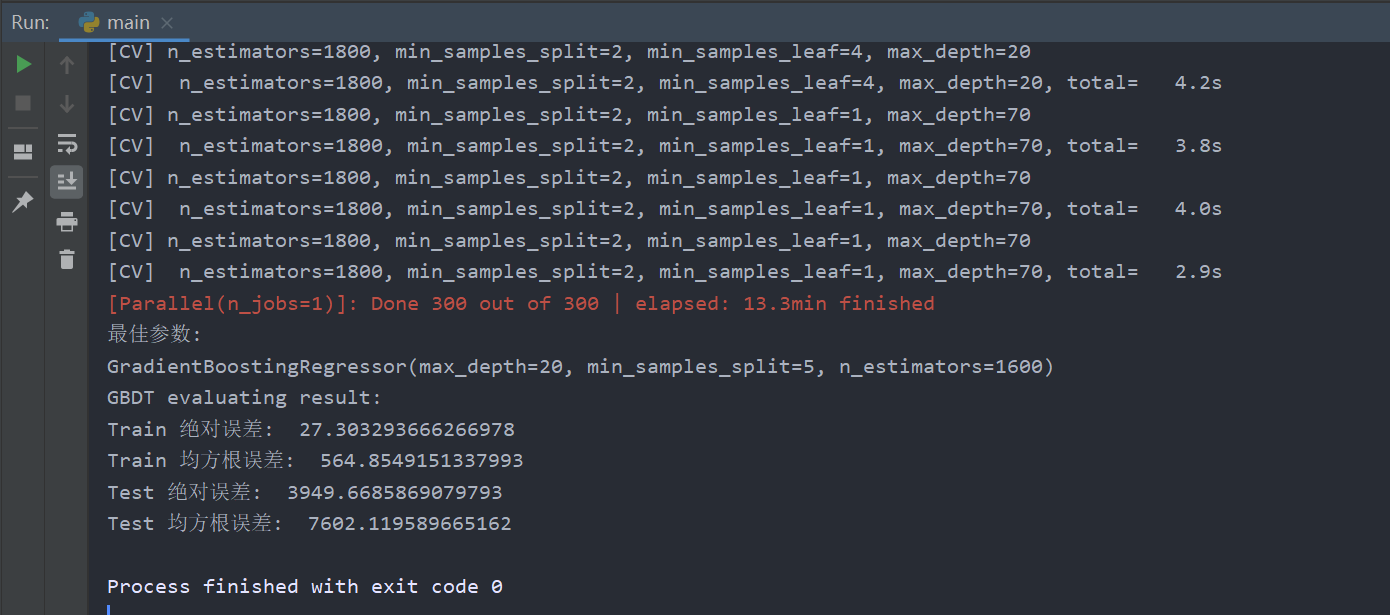
遇到的问题三：手写实现线性回归，结果误差较大。

解决方法：对数据进行归一化。在使用线性回归时，数据需要进行归一化处理，才能保证准确率。因为数据做了归一化后，整个数据集上的梯度分布得到了改良，此时也可以适当调大学习率。

遇到的问题四：模型融合结果比不融合还低

解决方法：模型不满足“好而不同的特点”。不能直接使用直接平均法，根据模型的性能排名，为模型赋予一定的权重，按加权平均的方式，并对模型部分异常值进行剔除，不参与模型融合。

遇到的问题五：调参时出现过拟合

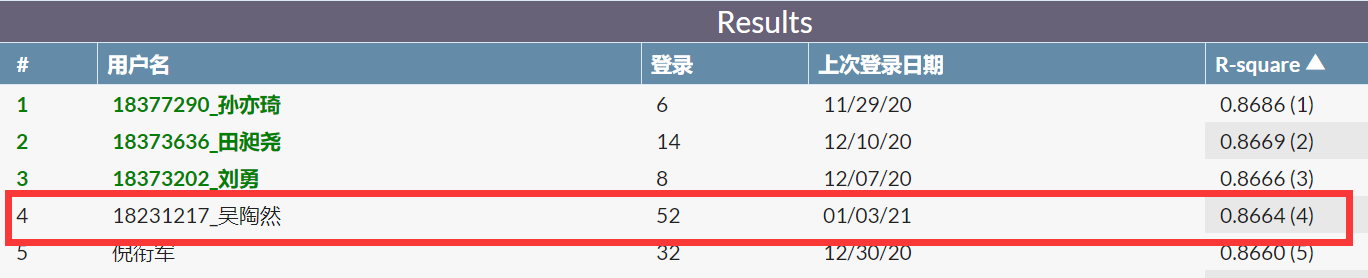


解决方法：模型预测结果里，训练集的误差很小很小，而测试集的误差很大很大，甚至在训练集误差的十倍以上时，大概率为过拟合。此时需要注意重新调节参数，特别是学习率等参数。决策树容易出现过拟合的现象，而随机森林和GBDT等不容易过拟合。

5. 评测分数和排名

截止2021年1月3日晚20:00，最终评分为**0.8664**，排名为**第4名**。





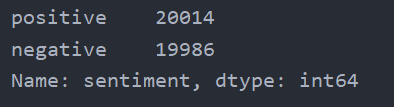
二. IMDB评论情感判断

1. 方法介绍

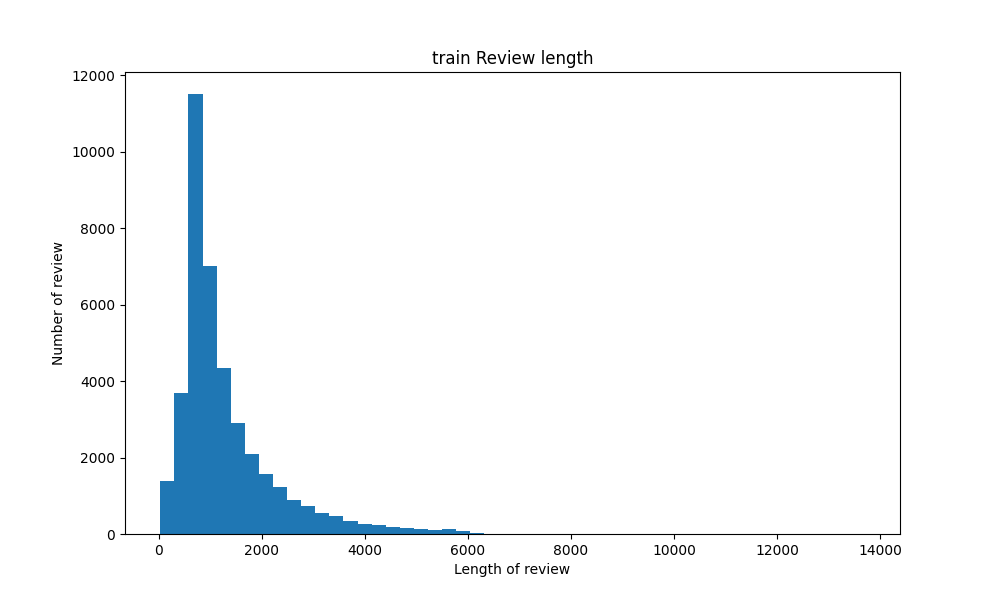
使用stopwords、WordNetLemmatizer、Tokenizer进行数据预处理。使用Keras中的Sequential模型进行训练，共包含Embedding层、LSTM层、池化层、激活函数为relu的隐藏层和激活函数为sigmoid层。

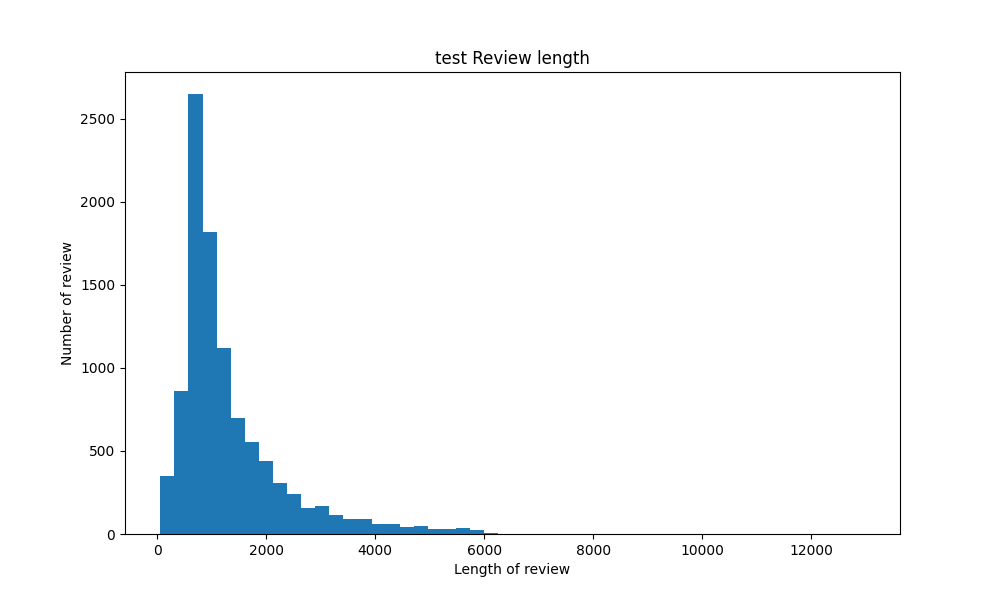
2. 数据处理、方法实现细节和模型结构

通过pd.read\_csv读入数据后，对’sentiment’特征调用value\_counts()方法观察得知数据positive和negative大概分布均衡。



同时训练集和测试集的大部分评论的长度均在6000以下。





使用LabelEncoder函数将数据集中的positive转换为1，negative转换为0，便于模型的训练。通过观察数据集可以得知，评论中存在<br /><br />等html字符，需要将其剔除。可使用正则表达式匹配，直接将其删去。并将所有评论字符转换为小写。

数据还需进行stop words（停用词）的预处理，这些词应用十分广泛、对情感分析并无太大意义，可能自身并无明确的意义，例如语气助词、副词、介词、连接词等。为节省存储空间和提高搜索效率，在进行建模时需要忽略stopwords。故引入nltk.corpus库中的stopword，删除评论中的所有停用字符。

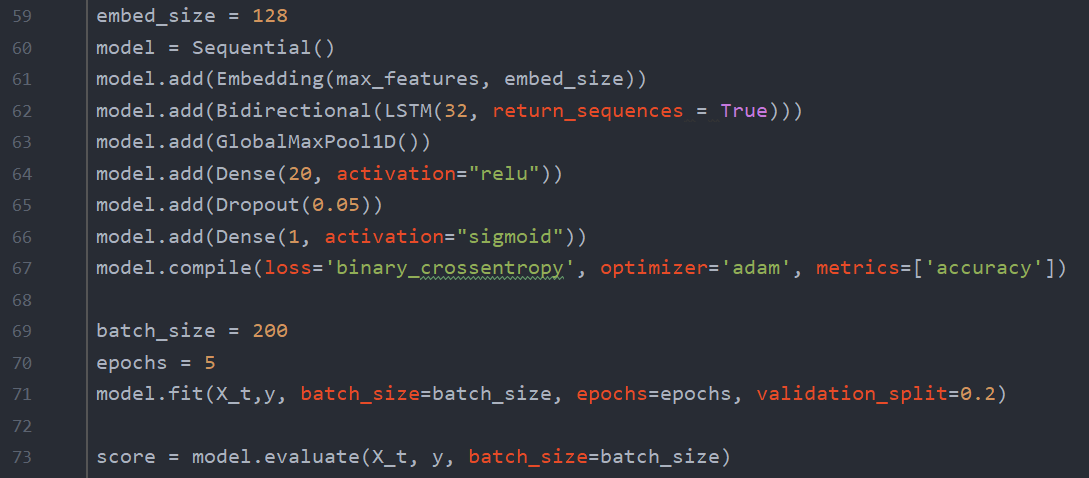
由于单词存在不同形式，例如have单词可能有having或has等形式，故使用nltk.stem中的WordNetLemmatizer进行词干提取。

随后使用Keras的分词器Tokenizer，将文本转换为序列。由于通过分析数据得知，训练集和测试集的评论大部分在6000以下，故将Tokenizer的num\_words属性设置为6000。使用pad\_sequences对序列进行预处理，maxlen参数设置为None。

建立Sequential模型，使用Embedding加入Embedding层，加入LSTM网络，使用GlobalMaxPool1D()加入池化层，加入一层激活函数为relu的隐藏层，维数为20。并引入Dropout()防止过拟合，参数设置为0.05，最终输出层激活函数为sigmoid，维数为1。

使用compile方法和fit方法对模型进行训练，其中loss参数设置为’binary\_crossentropy’，optimizer参数设置为’adam’， metrics参数设置为['accuracy']，batch\_size 设置为200，共迭代5次。

最终，该模型包含Embedding层、LSTM层、池化层、激活函数为relu的隐藏层和sigmoid层。参数设置如下：



由于模型训练较为耗时，故使用Kera的save方法将训练得到的模型进行保存。对测试集进行相同的测试处理后，使用predict方法进行训练以后即可得到结果。

3. 调参过程

由于该问题中需要调节的参数很少，且进行极少的调整后就能得到不错的结果，故直接手动尝试调参。

Tokenizer方法的num\_words，由于通过数据可视化可以直观感受到数据集中大部分评论的长度均在6000以下，故num\_words直接设置为6000.

pad\_sequences里，为maxlen设置一定限制能大幅提高训练速度，根据观察处理后的数据可知，将maxlen设置为150较为合适。但由于为了追求极致的正确率，此处将maxlen设置为None。

embed\_size最终设置为128。初始时忽视了batch\_size参数的设置，以至于在训练过程中所有的训练数据直接输入到网络，导致泛化能力差，计算量庞大。最终batch\_size设置为200，epochs设置为5。

4. 遇到的问题和解决方法

遇到的问题一：模型训练效率过低，训练一次需要耗费几天的时间

解决方法：引入stopwords停用词集，删除停用词，减少不相关单词的空间占用和时间占用；加入batch\_size参数限制，初始时设置为100，调参后更改为200；加入maxlen参数限制，将过长的序列截断。优化后效率大幅提升。

遇到的问题二：nltk里的stopwords始终无法正确调用和配置。

解决方法：不通过pip install或者pycharm的环境按照nltk，而是在github上找到完整的nltk数据包，根据调用时的错误提示安装在合适的位置即可。

5. 评测分数和排名

截止1月3日下午17:00，最终评分为**0.9296**，排名为**第8名**。



