**华中科技大学**

**计算机科学与技术学院**

**《机器学习》结课报告**



专 业： 计算机科学与技术

班 级： 计算机2011班

学 号： U202010755

姓 名： 路昊东

成 绩：

指导教师： 何琨

**完成日期： 2022年 7 月 5 日**

目录

[1. 实验题目: 商品销量智能预测挑战赛 2](#_Toc8890)

[2. 实验要求 2](#_Toc21952)

[2.1 实验任务 2](#_Toc8413)

[2.2 数据说明 2](#_Toc2475)

[2.3 评测标准 3](#_Toc19647)

[3. 算法设计 3](#_Toc14139)

[3.1 数据处理 3](#_Toc6158)

[3.2 模型算法 4](#_Toc8814)

[3.3 模型融合 5](#_Toc5043)

[4. 实验环境与平台 5](#_Toc2532)

[5. 实现与分析 5](#_Toc6080)

[5.1 数据处理 5](#_Toc29076)

[5.2 模型算法 8](#_Toc7120)

[5.3 模型融合思路 10](#_Toc28705)

[6. 实验结果 10](#_Toc19343)

[7. 个人体会 10](#_Toc31703)

[7.1 实验感想 10](#_Toc55)

[7.2 有待提升 11](#_Toc23708)

[7.3 个人建议 11](#_Toc20299)

# 实验题目: 商品销量智能预测挑战赛

随着企业持续产生的商品销量，其数据对于自身营销规划、市场分析、物流规划都有重要意义。但是销量预测的影响因素繁多，传统的基于统计的计量模型，比如时间序列模型等由于对现实的假设情况过多，导致预测结果较差。因此需要更加优秀的智能AI算法，以提高预测的准确性，从而助力企业降低库存成本、缩短交货周期、提高企业抗风险能力。

# 2. 实验要求

## 2.1 实验任务

本次大赛提供了商品销量历史数据作为训练样本，参赛选手需基于提供的样本构建模型，预测商品未来三个月的销售量。

## 2.2 数据说明

本次比赛为参赛选手提供了2类数据：商品历史销量数据和商品月订单数据。商品历史需求销量数据提供了商品编码、日期、是否促销、商品销售量。商品月订单数据提供了商品编码、商品类型、月份、订单数量、商品月初和月末库存量。（label空值的含义表示该商品当天无销量），如表1所示：

表一 数据说明



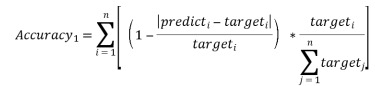
训练集为2018年2月1日至2020年12月31日的若干商品历史销量数据和订单数据;

测试集为2021年1月至3月的销量数据。

## 2.3 评测标准

根据整理，我归纳采用评价指标为：

**（1）**计算每个月商品预测准确率Accuracy1、Accuracy2、Accuracy3，其中Accuracy2、Accuracy3的计算公式和Accuracy1一样：

 **(1)**

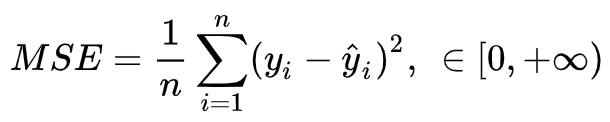
其中：predicti是商品i销量预测值，targeti是商品i销量真实值。

**（2）**计算累计所有预测月份的商品平均准确率：

 **(2)**

**（3）MSE**：

MSE(均方误差)函数一般用来检测模型的预测值和真实值之间的偏差。MSE是真实值与预测值的差值的平方然后求和平均。通过平方的形式便于求导,所以常被用作线性回归的损失函数。

 **(3)**

# 3. 算法设计

## 3.1 数据处理

* **数据对齐**方面，由于需求数据集为天级别，而订单数据为月级别，要根据一些特征字段进行唯一匹配将两组数据的字段合并起来；
* **缺失值处理**方面，当该月未销售该商品时，数据表中为空，而空值在被pandas读取时便被读为NaN，故采取删除变量的处理方法；
* **数值型替换**方面，数据中type字段为非数值型，无法在模型中直接使用，因此将其替换为0，1，2等数值；
* **归一化**方面，数据中不同字段的值域范围不同，直接输入模型对最终结果影响不同，因此将数据映射到[0,1]区间，采取最大-最小规范化方法；
* **数据标签拆分**方面，数据与标签分别表示为模型的输入与输出，本任务为根据历史数据预测未来数据，因此，拆分时选择过去8个月的数据作为输入，预测未来3个月的销量输出。
* **数据划分**方面，通过五折交叉验证的方式划分训练测试集。

## 3.2 模型算法

* 本次比赛**开始**使用**XGBoost树模型**:

它是基于预排序方法的[决策树](https://so.csdn.net/so/search?q=%E5%86%B3%E7%AD%96%E6%A0%91&spm=1001.2101.3001.7020" \t "https://blog.csdn.net/weixin_46649052/article/details/_blank)算法。首先，对所有特征都按照特征的数值进行预排序。其次，在遍历分割点的时候用O(#data)的代价找到一个特征上的最好分割点。最后，在找到一个特征的最好分割点后，将数据分裂成左右子节点。通过精确地找到分割点。然而很遗憾，得分不是很理想（第五节分析一下原因）。

* 于是**之后**更换了下思路，采用了个人稍微熟悉一点的**LSTM单模型**:

LSTM即长短时记忆网络（Long Short Term Memory，简称LSTM）模型，本质上是一种特定形式的循环神经网络（Recurrent Neural Network，简称RNN)。LSTM模型在RNN模型的基础上通过增加门限（Gates）解决RNN短期记忆的问题，使得循环神经网络能够真正有效地利用长距离的时序信息。LSTM在RNN的基础结构上增加了输入门限（Input Gate）、输出门限（Output Gate）、遗忘门限（Forget Gate）３个逻辑控制单元，且各自连接到了一个乘法元件上，通过设定神经网络的记忆单元与其他部分连接的边缘处的权值控制信息流的输入、输出以及细胞单元（Memory cell）的状态，如图3.1所示。相关部件的描述如下：

Input Gate：控制信息是否流入Memory cell中。

Forget Gate：控制上一时刻Memory cell中的信息是否积累到当前时刻Memory cell中。

Output Gate：控制当前时刻Memory cell中的信息是否流入当前隐藏状态中。

cell：记忆单元，表示神经元状态的记忆，使得LSTM单元有保存、读取、重置和更新长距离历史信息的能力。

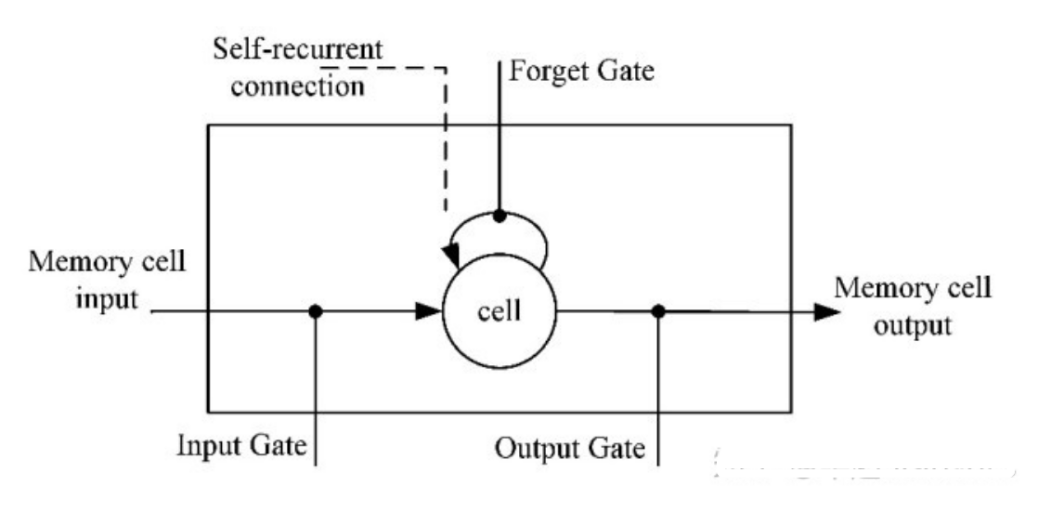
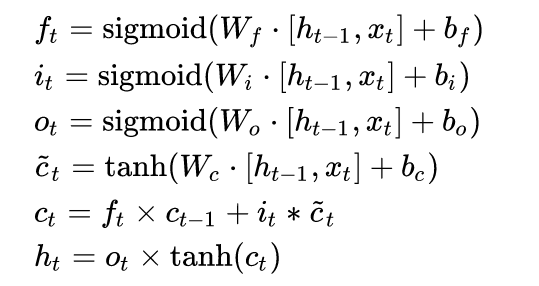


图3.1 LSTM神经网络图示

在t时刻，LSTM神经网络定义的公式如下：



* 当然我还了解到诸如**LightGBM**之类的树模型算法，其对经典的GBDT工具XGBoost采取了优化，由于时间关系只是浅浅做了一下尝试，发现得分和之前的XGBoost差不多，遂放弃，此处也不再赘述我对于LightGBM的体会。

## 3.3 模型融合

本任务由于是回归任务，采用LSM单模型，因此融合时将五折交叉验证中各模型的输出求平均作为最终的预测输出。

**不过这里我仍旧给出一个模型融合的思路，采用Stacking的思想，在第五节详细描述。**

# 4. 实验环境与平台

* 硬件部分：CPU：Intel CORE i9-9900K 3.6GHz，处理器规格：8核16线程；显卡：2080Ti；显存：11G；内存：32G。
* 实验环境配置：操作系统：Ubuntu20.04；基础环境：CUDA 11.3、CUDNN 7.9；使用PyTorch1.10的编程框架来进行深度学习编程；采用Python3.8.10版来完成代码编译。

# 5. 实现与分析

## 5.1 数据处理

在这个偏数据挖掘的比赛中，数据处理是重中之重。根据对数据的理解抽象归纳出有用的特征从而优化数据处理后对个人提分0.2左右，下面是主要方法：

* **数据对齐**

针对需求数据集（天级别）和订单数据（月级别）的不一致，且最终任务对月级别回归，因此对需求数据进行合并：其中主要包含每天是否为促销日、每天的销量，对促销日来讲，只考虑到每个月中促销日的比例对销量的影响，因此促销日每个月的促销日取平均为每个月的促销比例；月销量则是每天销量之和，进行求和操作。对这两个字段进行操作之后得到月级别的字段，其中促销日的范围为0~1，月销量则为大于等于0。需求数据与订单数据均为月级别数据后可以进行合并，主要根据product\_id、year、month三个字段进行唯一匹配将两组数据的字段合并起来，得到合并后的数据，如图5.1所示：

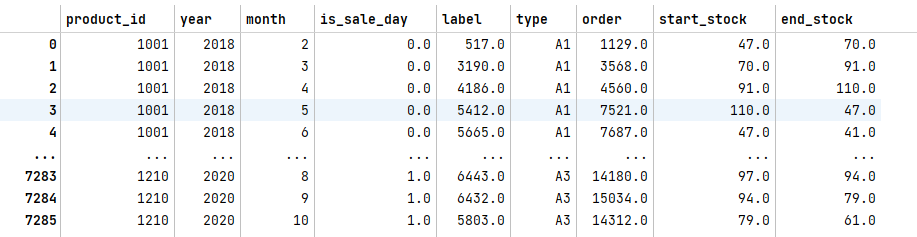


图5.1 数据合并

* **缺失值处理**

商品销量为0时在数据表中为空，被pandas读取时便被读为NaN，而0其实对模型的销量预测也并没有作用（变量的缺失率较高，覆盖率较低，且重要性较低），因此我将具有空值数据全部丢弃，代码如下：

***# 缺失值,缺失行label均为0*null\_index = df.isnull().any(axis=1)==*True*df[null\_index]['label'].sum()  
df = df[~null\_index]**

* **数值型替换**

数据中type字段为非数值型，无法在模型中直接使用，因此将其替换为数值型，type中主要为A1、A2、A3三种，因此将其分别替换为0，1，2数值。

**df = df.replace('A1', 1)  
df = df.replace('A2', 2)  
df = df.replace('A3', 3)**

* **归一化**

这里采用最大 - 最小规范化。主要针对label, order, start\_stock, end\_stock四个字段。需要注意的是，在进行归一化时，由于start\_stock, end\_stock具有相关性，因此计算归一化时一同考虑计算，即最大值最小值相同，最终归一化为0-1之间的数值，计算公式为：

（4）

* **数据标签拆分**

数据与标签分别表示为模型的输入与输出，本任务为根据历史数据预测未来数据，因此，拆分时选择过去8个月的数据作为输入，预测未来3个月的销量输出。需要注意的是，未来三个月除了销量的其他字段也已经给出，因此将未来三个月的字段也加入进来，其中销量字段设置为0。另外，划分数据标签时根据product\_id进行划分，product\_id不作为变量输入训练预测，代码如下（有省略）：

**group\_df = df.groupby('product\_id')  
norm\_feat = ['label', 'order', 'start\_stock', 'end\_stock']  
feat = ['month', 'type', 'is\_sale\_day', 'label', 'order', 'start\_stock', 'end\_stock']**

……

***for* key, value *in* group\_df:  
 group\_df\_value[key] = copy.deepcopy(value)  
 maxvalue[key] = value[norm\_feat].max()  
 minvalue[key] = value[norm\_feat].min()  
 maxvalue[key]['start\_stock'] = *max*(maxvalue[key]['start\_stock'], maxvalue[key]['end\_stock'])**

……

**value[norm\_feat] = (value[norm\_feat]-value[norm\_feat].min() + 1e-5) / (value[norm\_feat].max() - value[norm\_feat].min() + 1e-5)  
id\_xvalue, id\_y\_value = [], []  
 *for* i *in range*(in\_steps, *len*(value) - out\_steps):  
 *if* value['label'][i-8:i-4].values.tolist() == [0,0,0,0]:  
 *continue***

……

**ids.append(key)  
 id\_x\_values[key] = id\_xvalue  
 id\_y\_values[key] = id\_y\_value**

* **数据划分**

模型训练需要划分训练测试集，为充分评估模型性能，通过五折交叉验证的方式划分训练测试集（通常比按6:2:2划分训练集验证集测试集上的结果要好）：

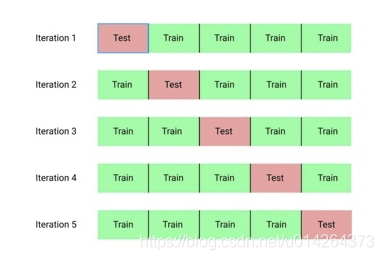


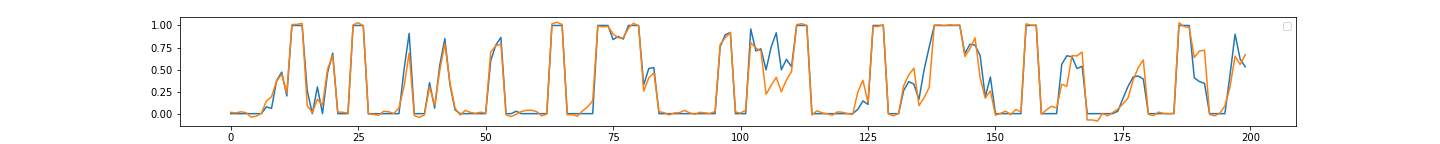
图5.2 五折交叉验证图示

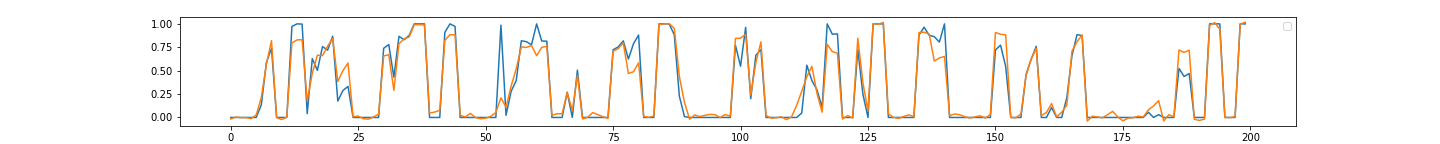
具体来讲，把数据平均分成5等份，每次实验拿一份做测试，其余用做训练。实验5次求平均值。如上图，第一次实验拿第一份做测试集，其余作为训练集。第二次实验拿第二份做测试集，其余做训练集。依此类推，能够得到五次不同的模型（有省略）：

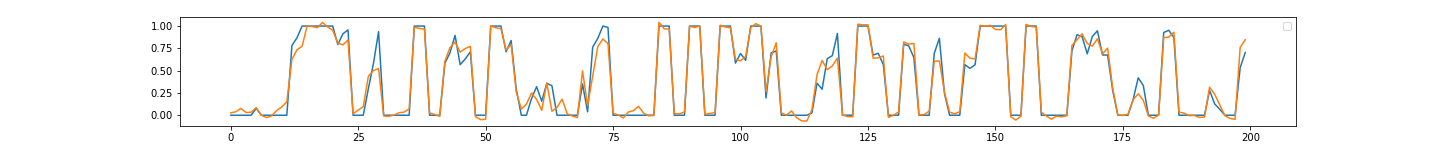
***# 训练测试划分  
import* random  
n\_split = 5  
random.shuffle(ids)  
x\_train, y\_train, x\_test, y\_test = {}, {}, {}, {}  
ids\_map = []  
*for* tmpi, i *in enumerate*(*range*(0, *len*(ids), *len*(ids)//5)):  
 test\_ids = ids[i: i + *len*(ids) // 5]  
 train\_ids = *list*(*set*(ids) - *set*(test\_ids))  
 ids\_map.append([train\_ids, test\_ids])  
 x\_train[tmpi] = []  
 y\_train[tmpi] = []  
 *list*(*map*(*lambda* x: x\_train[tmpi].extend(id\_x\_values[x]), train\_ids))  
 *list*(*map*(*lambda* x: y\_train[tmpi].extend(id\_y\_values[x]), train\_ids))  
 x\_test[tmpi] = []**

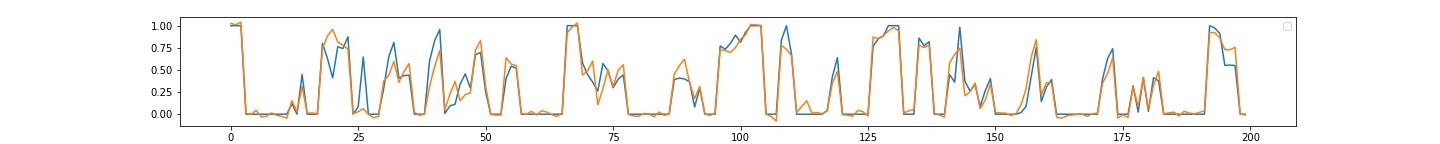
……

五折交叉验证的验证集结果如图5.3所示：









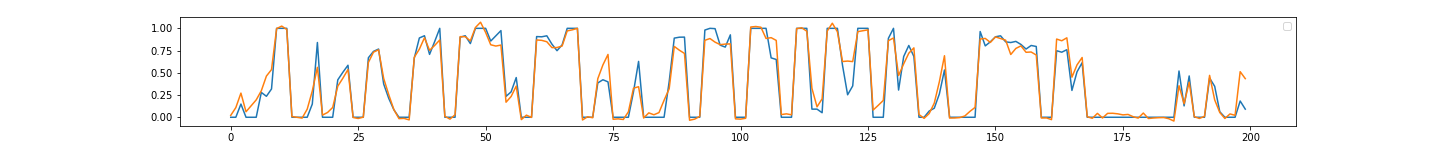


图5.3 五折交叉验证结果

## 5.2 模型算法

这是另外一个影响得分的因素（更换模型之后直接从0.3+跳跃到0.6+）最终使用LSTM模型，开始使用了XGBoost模型，可能是个人问题（调参）导致XGBoost得分不高。

这里不妨首先分析一下XGBoost模型的不足：首先，**空间**消耗大。这样的算法需要保存数据的特征值，还保存了特征排序的结果（例如，为了后续快速的计算分割点，保存了排序后的索引），这就需要消耗训练数据两倍的内存。其次，**时间**上也有较大的开销，在遍历每一个分割点的时候，都需要进行分裂增益的计算，消耗的代价大。**最后**，感觉对cache优化不友好。预排序后，特征对梯度的访问是一种随机访问，并且不同的特征访问的顺序不一样，无法优化cache。同时，在每一层长树的时候，需要随机访问一个行索引到叶子索引的数组，并且不同特征访问的顺序也不一样，也会造成一定程度上的cache miss。

于是来到了一个经典问题：DL模型vs GBDT模型。对比这两种模型我认为决定性因素之一是**数据可用性**：LSTM 是深度模型，通常与 CNN 和密集层结合使用；因此，它们需要大量数据进行训练。另一方面，GBT 模型用较小的训练集来饱和它们的泛化。一句话而言， GBM 是总体上最安全的赌注，但大多数时候，出色的结果将来自深层网络。

当然，这里我的 LSTM 比 XGBoost 表现得更好，可能是我没有认真调整它们的超参数，毫无疑问，一个好的超参数优化算法格外重要。（不过我们不能否认还要在泛化性和准确性之间进行权衡，其实终究要特殊问题特殊分析——数据集类型是什么、所有值是否连续等等都是模型选取的影响因素）。

总之，这里使用LSTM，网络模型搭建如下，输入向量的[维度](https://so.csdn.net/so/search?q=%E7%BB%B4%E5%BA%A6&spm=1001.2101.3001.7020" \t "https://blog.csdn.net/comli_cn/article/details/_blank)input\_size、隐藏层向量的维度hidden\_size、循环层的数量num\_layers分别为7、32、2，只需采用单向的LSTM网络，模型搭建如下（代码有省略）：

***class* LSTM(nn.Module):  
 *def \_\_init\_\_*(*self*, input\_size, hidden\_size, num\_layers, output\_size, batch\_size):  
 *super*().*\_\_init\_\_*()  
 *self*.input\_size = input\_size**

……

***self*.num\_directions = 1 *# 单向LSTM*  
 *self*.lstm = nn.LSTM(*self*.input\_size, *self*.hidden\_size, *self*.num\_layers, batch\_first = *True*)  
 *self*.linear = nn.Linear(*self*.hidden\_size, *self*.output\_size)  
 *def* forward(*self*, input\_seq):  
 *# 两个初始化* h\_0 = torch.randn(*self*.num\_directions \* *self*.num\_layers, *self*.batch\_size, *self*.hidden\_size).to(device)  
 c\_0 = torch.randn(*self*.num\_directions \* *self*.num\_layers, *self*.batch\_size, *self*.hidden\_size).to(device)  
 *# 前向计算* output, \_ = *self*.lstm(input\_seq, (h\_0, c\_0))  
 pred = *self*.linear(output)  
 pred = pred[:, -1, :]  
  *return* pred**

搭建模型之后就开始训练，首先定义模型优化器、损失函数等参数：model 、train\_loader、loss\_function、schedule，训练过程如下（有省略）：

***for* epoch *in* pbar:  
 train\_loss = []  
 *for* seq, label *in* train\_loader:  
 seq = seq.to(device)  
 label = label.to(device)  
 y\_pred = model(seq)  
 loss = loss\_function(y\_pred, label)**

……

**optimizer.step()  
 scheduler.step()  
 *# validation* val\_loss = get\_val\_loss(model)  
 *if* epoch > min\_epochs *and* val\_loss < min\_val\_loss:  
 min\_val\_loss = val\_loss  
 best\_model = copy.deepcopy(model)**

……

**model.train()**

……

## 5.3 模型融合思路

我们知道不同的模型原理和所得结果之间是存在差异的。所以这里可以借鉴**Stacking**的思想（就是将一系列模型的输出结果作为新特征输入到其他模型，细节不必复述），融合LightGBM、XGBoost以及[LSTM](https://so.csdn.net/so/search?q=LSTM&spm=1001.2101.3001.7020" \t "https://blog.csdn.net/keypig_zz/article/details/_blank)三个模型（这里参考了一些博客文章）。**例如，不妨分为三层**：使用XGBoost-1对特征组合F1进行学习，得到XGBoost-1的预测结果，该结果作为新特征，加入特征组合F2和F3中，分别作为第二层LightGBM-1 和 LightGBM-2的输入特征，LightGBM-1的结果再次作为新特征，加入特征组合F4中，作为第三层XGBoost-2的输入特征，同时第三层包含一个LSTM模型，该模型使用特征组合F5训练，第二层LightGBM-2的结果则与第三层XGBoost-2,LSTM的预测结果进行加权融合作为最终结果。

不难得到，三种模型中其中两种是树模型，LSTM为神经网络模型。这两类模型原理相差较大，且学习能力都很强，同时产生的结果相关性较低，融合有利于提高预测准确性。当然“实践出真知”，此想法后续还要继续验证。

# 6. 实验结果

本次比赛总共提交7次数据，其中影响因素最大的是模型优化以及数据处理优化，截至7月6日最终排名为80名，得分0.60173，如图6.1所示：



图6.1 比赛结果截图

# 7. 个人体会

## 实验过程感想：

本次实验算是对于课程以及平时自学过程的一个答卷，虽由于时间和个人精力的原因并没有很完善，但的确也算是在规定时间尽力完成。下面就是我的一些总结和我这个菜鸟的心路历程吧。

这是我第一次接触此类编程比赛，首先的难点就在于**分析题目**，感觉大体上的机器学习问题类型就是回归问题和分类问题。

其次也是比较折磨的一点就是对于**数据的分析和处理**。所谓数据挖掘，当然是要从数据中去挖掘我们想要的东西，归纳总结，我认为在观察数据的过程中思考以下几个**问题**：数据应该怎么清洗和处理才是合理的？根据数据的类型可以挖掘怎样的特征？数据中的哪些特征会对标签的预测有帮助？至于**统计分析**，对于数值类变量，我们可以得到一系列统计量，用[pandas](https://link.zhihu.com/?target=http://pandas.pydata.org/)可以方便地完成。我们知道，刚拿到手的数据会出现噪声，缺失，脏乱等现象，所以需要对数据进行**清洗与加工**，从而方便进行后续的工作。这次比赛的数值型变量，就需要处理缺失值，异常值等。此外还可以数据归一化，提高模型的精度。此外对于数据，我们应该尽可能多地抽取**特征**，我认为只要某个特征对解决问题有帮助，它就可以成为一个特征。特征抽取需要不断迭代，是最为烧脑的环节，但是的确值得耗费大量的时间。至于发现特征，只是盯着[数据集](https://so.csdn.net/so/search?q=%E6%95%B0%E6%8D%AE%E9%9B%86&spm=1001.2101.3001.7020" \t "https://blog.csdn.net/u010899985/article/details/_blank)肯定是不行的。对于本次比赛的数值型变量，可以通过线性组合、多项式组合来发现新的特征，也是一个很遗憾的点，由于精力和水平问题，我找到的特征有限，所以分数也有限。

后面就是**炼丹环节**，这也是让我既痛苦又快乐的一点。很多模型都有回归和分类两种版本。本次比赛我首先就一种机器学习模型**XGBoost**进行了尝试，但是可能还是由于对于模型的原理没有完全掌握，调参环节出了问题，导致效果不佳，于是又使用深度学习工具PyTorch定制了自己的DL模型，事情变的有趣很多，结果也得到了提升。然而我还是对于大家口口相传的神器模型心向往之，于是又尝试了LightGBM,效果依然不如之前的DL模型，所以最终就只是使用了单独的**LSTM**模型，这也算是我这次比赛的一大遗憾吧。当然最大的遗憾在于**Ensemble**环节。Feature决定了模型效果的上限，而Ensemble就是让我们更接近这个上限，辛辛苦苦学到也打算使用的stacking方法没有时间使用，只是提出了一个较为可行的思路，当然这也是我之后努力的方向。

总之，这门课程的学问博大精深，这类比赛也蛮有趣味，后续如有时间我当然会继续尝试、继续实践！

## 有待提升：

* **特征工程**方面，关于一些特征的抽取以及选择应进一步加强，争取能够抽取到更多的有效特征同时避免过拟合问题（继续使用相关系数/训练模型筛选特征）。
* **模型的使用和融合**方面，继续加强对一些机器学习模型的学习以及调参方法的学习，同时多尝试一下模型融合的方法，多做实践。

## 个人建议：

真诚希望下次老师提早布置一下相关任务，本次由于同时段考试过多，时间比较仓促，无法集中精力进行课外学习和工程的优化，大都是靠着平时自学积累完成。