目录

[1. 问题描述 1](#_Toc2027)

[2. 设计简要描述 1](#_Toc1935)

[3. 概要设计 2](#_Toc29175)

[4. 程序分析 3](#_Toc696)

[4.1 导入必要库 3](#_Toc31930)

[4.2 训练集数据处理 4](#_Toc30039)

[4.3 将train data划分为train\_x和train\_y 6](#_Toc21828)

[4.4 处理test data 8](#_Toc7966)

[4.5 添加偏置（更好的拟合数据集） 10](#_Toc28637)

[4.6 数据导出 10](#_Toc28970)

[4.7 线性模型的构成 11](#_Toc22357)

[4.7.1 库文件导入 11](#_Toc22665)

[4.7.2 线性模型的构建 12](#_Toc9215)

[4.8 调用模型完成预测 15](#_Toc21977)

[5. 结果分析 16](#_Toc16201)

[5.1 结果一 16](#_Toc4847)

[5.2 结果二 20](#_Toc335)

[6. 其他思路 21](#_Toc28328)

[7. 经验与体会 22](#_Toc24547)

# 问题描述

学会使用学习到的线性回归的知识，手动使用adagrad梯度下降方法，通过给定的相关数据来完成PM2.5值的回归预测。

# 设计简要描述

（1）确定数据集

本次作业是使用某监控站的观测记录，分成train set 以及

test set，train set 是该站每个月前20天的所有资料，test set 是从该站剩下的资料中取样出来。即：

①Train.csv每个月前20天的完整资料。

②Test.csv从剩下资料中取样出连续的10小时为1笔数据，前9小时所有的观测数据当做feature，第10小时的PM2.5 当做answer，一共可以取出240笔不重复的test data，根据feature预测这些240笔的PM2.5.

本次作业需要完成的任务是通过test data，取出240笔不重复的test data，根据feature预测这些240笔的PM2.5.

# 概要设计

3.1 train data

（1）分析train data

train data指12个月，每个月前20天的所有数据；

每天包含24小时，每一个小时数据可以分为18种污染指标的具体数值：AMB\_TEMP, CH4, CO, NHMC, NO, NO2, NOx, O3, PM10, PM2.5, RAINFALL, RH, SO2, THC, WD\_HR, WIND\_DIREC, WIND\_SPEED, WS\_HR。

根据test data给出的提示，可以知道，大概是将每10小时作为一笔数据，我们通过每一笔数据中前九个小时的所有数据指标，完成第10个小时，PM2.5可能的值得预测。

我的理解：前9个小时每小时都有18个指标，总共18\*9=162个指标，每一个Feature都可影响第十个小时最后的PM2.5的值。

暂定思路：通过162个feature来预测一个PM2.5

1. 拆分train data

因为每一天都有24小时，数据集按照连续10小时为1笔的要求，

每个月则从前第一天到第20天可以共取得**471笔数据。**

**train\_x：**每一笔数据表示前九个小时162个污染数据指标

**train\_y：**每9小时的下一小时的PM2.5的值

3.2 test data拆分

test data提供的数据包含有9个小时的数据，一共包含240笔

最终预测出240个PM2.5数据

# 程序分析

* 1. **导入必要库**

|  |
| --- |
| # 引入必要的库函数  import numpy as np  import pandas as pd  import matplotlib.pyplot as plt  import gc # 垃圾回收  # 显示效果设置（可以无限制查看行和列）  pd.set\_option("display.max\_rows", None)  pd.set\_option("display.max\_columns", None) |

因为电脑内存只有8G，处理数据的时候超级卡，只好通过引用了**gc库**释放部分内存，进行开发

* 1. **训练集数据处理**

|  |
| --- |
| # 引入数据集合  train\_file = open('data/train.csv', encoding='big5')  # big5因为可能存在繁体字，使用big5编码格式以防出现乱码  train\_data = pd.read\_csv(train\_file)  # 删除无关列  train\_data = train\_data.drop(['Date', 'stations', 'observation'], axis=1)  # 脏数据处理，将所有的NR数据填充为0  train\_data = train\_data.replace('NR',0).T  # 把所有数据转化为float64类型  train\_data = train\_data.astype(float)  temp\_data = [] # 保存每一天的24小时的所有数据  for i in range(0,4319+1,18):  one\_day = train\_data.iloc[:,i:i+18]  one\_day.columns = list(range(1,19,1))  temp\_data.append(one\_day)  # 将每天的数据按顺序合并起来,一共得到240天，5760小时的数据  # 其中每个小时都包含18种污染物的数据指标  new\_train\_data = temp\_data[0]  temp\_data.pop(0)  for item in temp\_data:  new\_train\_data = new\_train\_data.append(item) |
| 图4.2.1 处理过后的train data数据  解释一下处理过后的train data：  总共18列，每一列分别代表了1种污染物  每一行则代表了污染物在对应的时间的指标值。  因此总共存在5760\*18个数据：  代表了18种污染物在总共5760个小时里的指标值 |

* 1. **将train data划分为train\_x和train\_y**

|  |
| --- |
| # 思路分析：  ①将连续九个小时的18种污染物数值（包括PM2.5）当做Featrues  即为train\_x 9\*18;  然后把其对应的第十个小时的PM2.5当做label 即为train\_y 1\*1  ②因为**时间需要连续**，所以必须按月划分，每个月按20天计算，总共480小时  则：可以拿到连续10小时数据的量为471次    '''  train\_x = pd.DataFrame() # 训练集Feature  train\_y = [] # 训练集Label  # 开始划分train\_x,train\_y  '''1. 最外层按月划分,间隔为480小时（一个月）  2. 第二层直接10小时一笔数据划分  '''  for i in range(0,5760,480): # 每个月数据的开始所在行共12个月  for j in range(i,i+471,1): # 每个月解析出总数据的笔数471笔  # 声明一个空的DataFrame列名为0-17，表示18种污染物  pd.DataFrame(columns=list(range(18)))  # 将每次循环的第一小时作为Feature的基底  one\_row\_feature = new\_train\_data[j:j+1].values  # 每9个小时当做一次train\_x  for n in range (j+1,j+9,1):  # one\_row\_feature保存了162个features的值  one\_row\_feature  = np.hstack((one\_row\_feature,new\_train\_data[n:n+1].values))  train\_x = pd.concat([train\_x,pd.DataFrame(one\_row\_feature)])  # 收集label:每组数据的第10个小时的PM2.5  train\_y.append(new\_train\_data.iloc[j+9:j+10,9:10]) |
| # 数据处理一下  train\_y = pd.concat(train\_y) # 将label转化为DataFrame  # 到这里已经完成了train\_x（162个数据，前9小时的18个污染物的指标值），train\_y（第10个小时的PM2.5的值）的提取 |

查看一下我们的train\_x,train\_y的值

可以看到行标为0~161，共162 ，这与9\*18 = 162是相符的

表示成功**将连续九个小时的18污染指标做成一条数据**

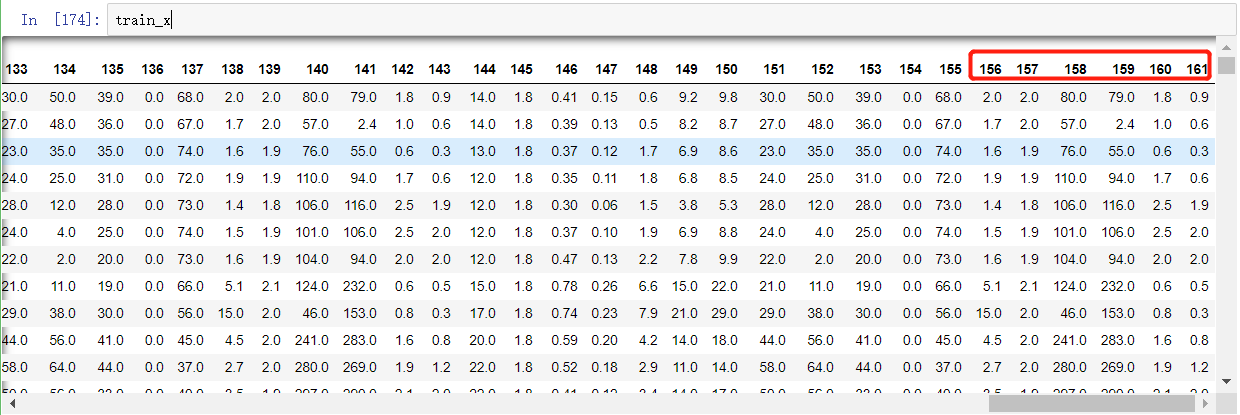


图4.3.1 train\_x的数据查看图

表示连续9小时后，**第十个小时对应的PM2.5的值**，当做训练Label

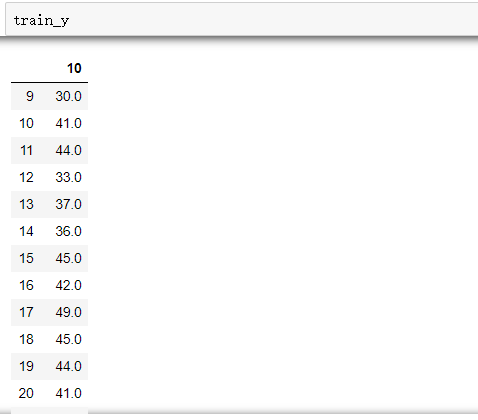


图4.3.2 train\_y 的数据查看图

|  |
| --- |
| del new\_train\_data  del temp\_data  del train\_data  gc.collect() # 内存回收 |

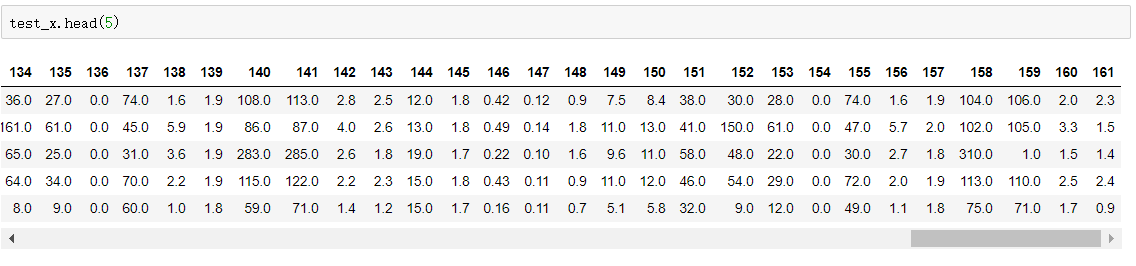
* 1. **处理test data**

|  |
| --- |
| with open("data/test.csv", encoding="big5") as test\_file:  test\_data = pd.read\_csv(test\_file,header=None)  test\_data = test\_data.replace("NR",0)  # 将数据修改为float类型  test\_data[[2,3,4,5,6,7,8,9,10]]= test\_data[[2,3,4,5,6,7,8,9,10]].astype(float)  # 测试集合总共天数 240天 id\_0 - id\_239.  days = len(test\_data[0].unique())  total\_rows = len(test\_data[0]) # 4320  # 测试集中包含9小时的数据，我们要通过这9小时的数据预测第十个小时的PM2.5  test= [] # 存放重构后的test数据  now\_row = 0 # 当前所在行数  for each\_day\_start\_row in range(0, total\_rows, 18):  # 每一行数据为连续九小时共162个污染指标的值  test.append([])  for i in range(2,11,1):  for j in range(each\_day\_start\_row, each\_day\_start\_row+18, 1):  test[each\_day\_start\_row//18].append(test\_data[i][j])  # test\_x收集完毕  test\_x = pd.DataFrame(test)  del test\_data # 释放资源  gc.collect() |

**查看test\_x的数据**

可以看到结果已经包含162列（index为0~161）

每一行数据表示一笔数据，用于预测第十个小时的PM2.5



* 1. **添加偏置（更好的拟合数据集）**

|  |
| --- |
| # 直接在数据集的第一列前面添加一列全1  train\_x = np.concatenate((np.ones((train\_x.shape[0],1)),train\_x), axis=1)  test\_x = np.concatenate((np.ones((test\_x.shape[0],1)),test\_x), axis=1) |

* 1. **数据导出**

因为内存实在不够，所以这里我将处理好后的数据统一导出结果

后面只关注模型的构建

|  |
| --- |
| pd.DataFrame(train\_x).to\_csv('./train\_x.csv', header=True)  pd.DataFrame(train\_y).to\_csv('./train\_y.csv', header=True)  pd.DataFrame(test\_x).to\_csv('./test\_x.csv', header=True) |

* 1. **线性模型的构成**

(这一部分仅实现了线性回归、正向传播和反向传播，可以不看)

(模型的实现可以参考代码demo)

4.7.1 库文件导入

|  |
| --- |
| # 由于换了一个文件，所以重新引入必要的库  import numpy as np  import pandas as pd  import matplotlib.pyplot as plt  # 设置numpy数组显示数据完全输出  np.set\_printoptions(threshold=None)  # 设置dataframe列数显示无限制  pd.set\_option("display.max\_columns", None) |

**4.7.2 线性模型的构建**

|  |
| --- |
| class MyLinearRegression():  '''手写模拟sklearn里面的LinearRegression  '''  def \_\_init\_\_(self):  pass  Def adagrad(self,train\_x,train\_y,w\_list,iteration=200000,lr=0.01, lambdaL2=0):  '''通过使用adagrad算法实现梯度下降    \_\_\_\_\_\_\_\_  parameters:  - train\_x:训练集的Features  - train\_y:训练集的Label  - w\_list：所有特征的weight集合  - lr：学习率  - iteration：迭代次数  \_\_\_\_\_\_\_\_  returns:  - w\_list: 返回模型训练完后的权重值vector  - cost\_list: 返回训练过程的cost，  '''  # 制造一个162长度的ndarray，保存每一个weight的grad更行过程中的平方和  sum\_grad = np.zeros((len(train\_x[0]),1))  cost\_list = [] # 保存历史损失值    # 开始梯度下降过程  for i in range(iteration):  y\_hat = np.dot(train\_x, w\_list) # 表示预测值  print(y\_hat.shape)  print(train\_y.shape)  # 表示预测值和估计值之间的差距  Lost = y\_hat - train\_y  # 1.计算损失  cost = np.sum(Lost \*\* 2)/len(train\_x)  cost\_list.append(cost)    # 2.计算梯度  grad = np.dot(train\_x.T, Lost)/len(train\_x) + lambdaL2 \* w\_list    # 根据adagrad算法计算每个参数的平方和  sum\_grad += grad\*\*2    # 更新参数  w\_list = w\_list - lr\*grad/np.sqrt(sum\_grad)  return w\_list,cost\_list      def predict(self, test\_x, w\_list):  '''根据传入的数据进行PM2.5的预测  \_\_\_\_\_\_\_\_  Parameters:  - test\_x：用于模型预测的测试数据  - w\_list：一组weight（由训练出的模型提供）  '''  predict\_result = np.dot(test\_x , w\_list) # 直接进行结果计算  return predict\_result # 返回预测结果 |

* 1. **调用模型完成预测**

|  |
| --- |
| # 读取数据  with open('../train\_x.csv',encoding='big5') as file1:  train\_x = pd.read\_csv(file1,index\_col=0)  with open('../test\_x.csv',encoding='big5') as file2:  test\_x = pd.read\_csv(file2,index\_col=0)  with open('../train\_y.csv',encoding='big5') as file3:  train\_y = pd.read\_csv(file3,index\_col=0)  train\_x = train\_x.values # 全部转化为ndarray形式方便矩阵乘法  train\_y = train\_y.values  test\_x = test\_x.values |
| # 1. 定义一个模型对象  model = MyLinearRegression()  # 2.模型训练  w\_list = np.zeros((len(train\_x[0]), 1))  cost\_list = []  w\_list,cost\_list = model.adagrad(train\_x,train\_y,w\_list, 500000, 1)  # 3.PM2.5预测  result = model.predict(test\_x,w\_list) |

# 结果分析

**5.1 结果一**

输出result结果并查看损失



可以很明显的看出，损失值迭代了几次，到达了下降到276.499的时候就一直不变，没有下降的趋势了。

（这样的结果肯定是不行的）

必定是特征工程出了问题，因此输出weight查看他们的变化，发现更新了几次之后就停止更新。

我特意查看了一下数据集，果然，是我在解析数据集的时候，出了问题

下图是第一种污染物的在每一天每一小时的指标含量。

可以发现从**第三波**开始，数据就都是一样的，显然后面数据都是一样的那就说明我的数据解析过程除了问题。所有数据都一样，那么训练出的模型误差就始终保持不变了，这也就是为什么我训练的结果十分不准的最大原因了。

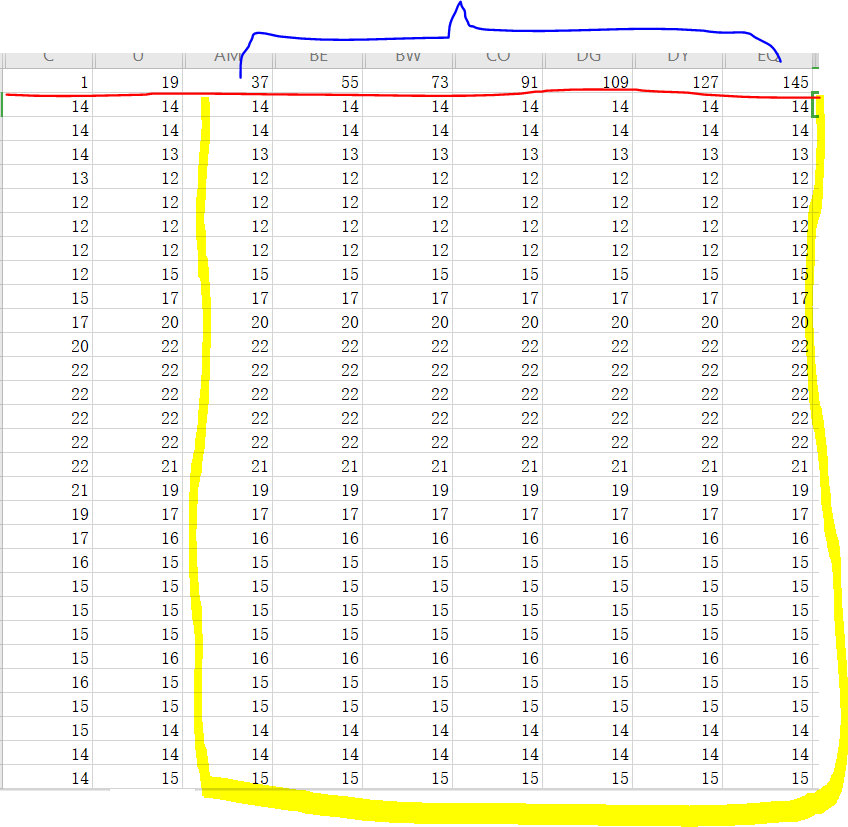


图5.1.1 错误分析1

**错误源头：**

经过逐一排查，终于在解析train\_x的语句中找到了错误

如下图：红色方框内部的一个循环

该循环是用来生成一笔（18\*9）数据，注意深红色下标的位置，我竟然使用了j+1，和j+2来作为下标从new\_train\_data中读取数据

显然该循环每次的j+1和j+2都是同一个值（**粗心大意**），所以才出现了图5.1.1数据的错误。

我真实想法是随着n的变化从new\_train\_data取数据，立马就知道如何修改。

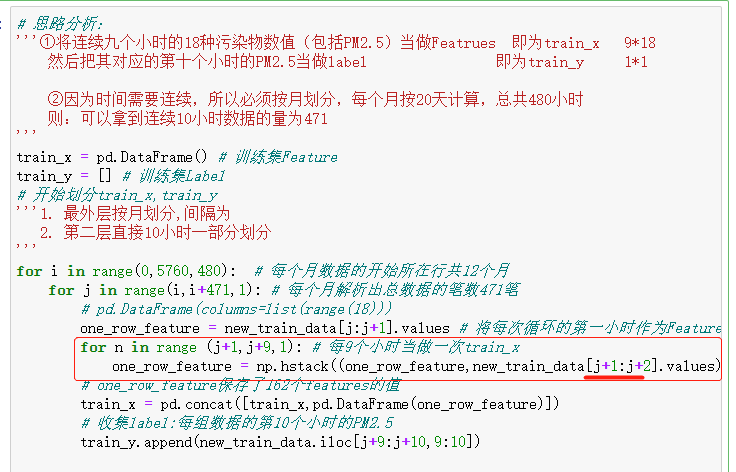


图5.1.2 错误分析2

**解决方案：**

完成如下图的下标替换即可完美解决

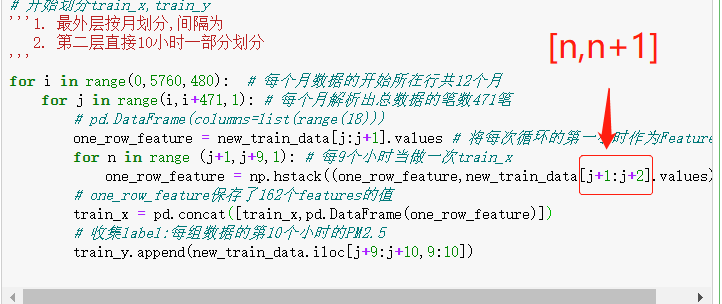


图5.1.3 错误解决方案

**5.2 结果二**

果然，这次的结果就十分正常，Loss值也在逐渐减少（这里的Loss是MSE）

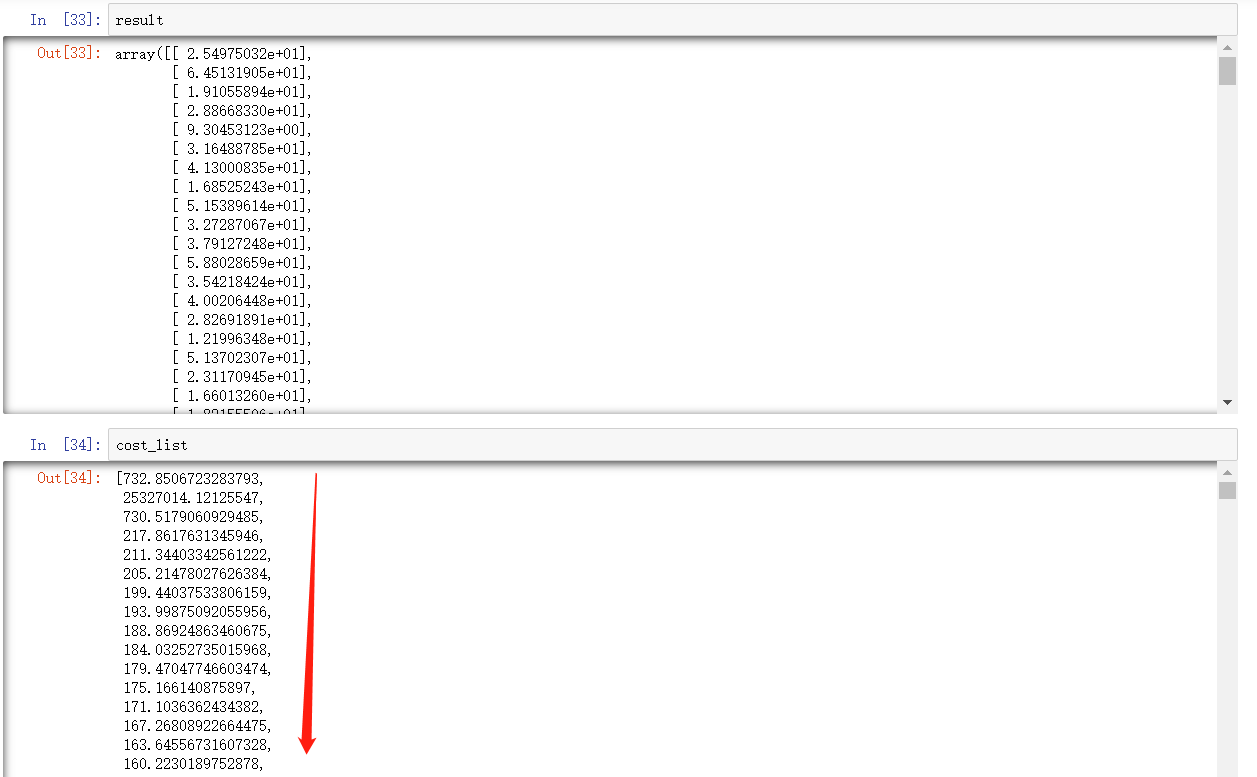


图5.2.1 正常结果图

如下图5.2.2则是模型训练的Loss最小值

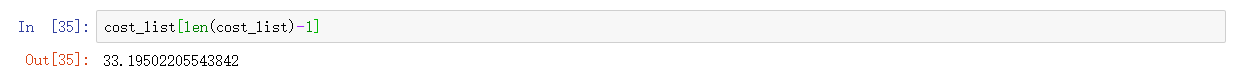


图5.2.2 Loss最小值

# 其他思路

**思路1：**特征不一定就选择这么多，因为前九个小时的所有污染指标都作为特征的话，光用来于预测一个PM2.5的话，特征就特别多（而且这些特征有的值的值大小范围很不相同，比如：NMHC为0.01~2.0左右，而WIND\_DIREC的范围就存在35~几百）。这样一来就很容易对模型产生过拟合，使得模型为了更好地贴合数据，使得误差增大。

**我的想法：**可以通过特征工程，借用matplotlib、seaborn可视化它们之间的关系，分析每种指标和PM2.5之间的关系，并查找出相关性的强弱，将相关性较弱的Feature去掉，然后在进行模型的训练，类似于去除一些杂音Feature，使得模型更好地关注与PM2.5相关的特征。

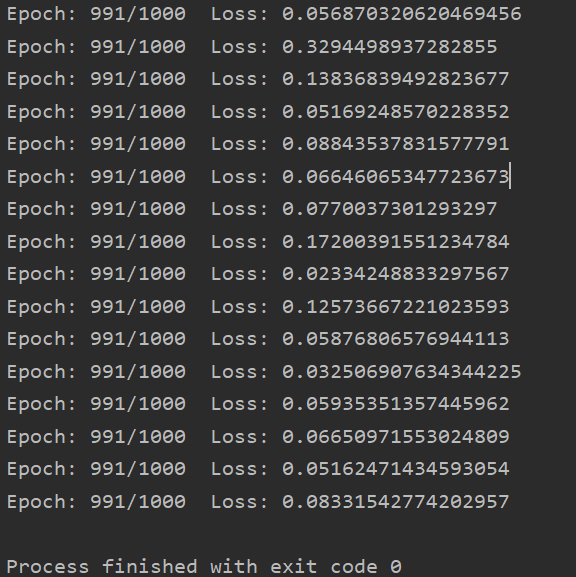
**思路2：**直接通过每一个小时的17污染物指标，将PM2.5作为Label，不关乎时间上的问题，来进行PM2.5的预测。这种PM2.5只能是一种相对来说较好的PM2.5值，一旦跟时间不能关联上，那么预测PM2.5的值就没什么意义，毕竟我们想要的是知道某个时间点的PM2.5的值，更有价值。

**思路3：**不关乎其他17种指标，只关注PM2.5随时间的变化，来进行预测PM2.5的值，但缺点是：忽视了太多相关特征，可能导致结果不好

**思路4：**就是本次已经实现了。

但是可以从结果看出，损失值只降低的不够。

在项目中的代码优化代码实现了损失降低到1以内。



# 

# 经验与体会

本次作业让我最大的体会就是：

① 数据处理对于机器学习或是以后的深度学习来说，都是最重要的，因为如果你的数据出了问题，模型写得再好，算法再好都是枉然。

② 在处理数据的过程中，发现数学得好，而且思路得清晰。

③ 其次，就是本次的作业核心部分了：模型的构建

之前都是直接调用sklearn里面的模型，根本没想过自己手动实现模型，所以总是觉得机器学习就这样，不会很难。但是通过本次作业，我能感觉到，首先懂得模型的原理是关键，模型的内部计算公式推导都得熟练；此外，最重要的就是细心了。

我在最后提供了一些其他思路，除了本次已经实现的思路4，第一种思路我还是比较认可的，但是还没来得及实现，不过有一点可以确定，只要数据处理的好，最后结果应该相差的不会太大。